Dynamic Overload Control for Distributed Call Processors Using the Neural Network Method

S. Wu*and K. Y. Michael Wong

Department of Physics,

The Hong Kong University of Science and Technology,

Clear Water Bay, Kowloon, Hong Kong.

phwusi@luc.ac.be and phkywong@usthk.ust.hk

April 2, 1998

Abstract

Overload control of call processors in telecom networks is used to protect the network of call processing computers from excessive load during traffic peaks, and involves techniques of predictive control with limited local information. Here we propose a neural network algorithm, in which a group of neural controllers are trained using examples generated by a globally optimal control method. Simulations show that the neural controllers have better performance than local control algorithms in both the throughput and the response to traffic upsurges. Compared with the centralized control algorithm, the neural control significantly decreases the computational time for making decisions and can be implemented in real time.

1 Introduction

In modern telecommunication systems, overload control is critical to guarantee good system performances of the call setup and disconnection processes. Overload events occur in heavy traffic, when the number of call setup jobs exceeds the capacity of call processing computers. These events, if left uncontrolled, will cause the system to break down and bring disasters to the network performance. Overload control is used to protect the limited system resources from excessive load, based on a throttling mechanism for new arriving requests. It is increasingly important with the emergence of Integrated Services Digital Networks (ISDN) [1], in which numerous customer services are provided and traffic is considerably higher.

This kind of control, which balances limited system resources on one hand and cumstomer requirements on the other, is widely encountered in telecommunications networks, e.g. in traffic routing [2], call admission control in ATM networks [3], channel assignment in wireless networks [4] and so on. These problems are often very difficult, since the traffic processes are stochastic and the degrees of freedom are large. It is difficult to find the optimal solution or the solution is too complex to be implemented.

In general, traffic control strategies can be implemented in two ways, local or centralized, according to the amount of information the control decisions depend on. Centralized control consists of one main networkwide controller which collects all the information through the signaling network. This is possible with the recent advances in the technology of the signaling networks, which enable a large amount of information to be tranferred instantly among system elements. It can make globally optimal decisions with the availability of networkwide information. However, it is often complex and time-consuming, and the work load of the signaling network is also high, rendering it impractical. Centralized control is also rather sensitive to network breakdown. On the other hand, local control makes decisions based on locally available information only.

It has the advantages of easy implementation and robustness to system breakdown. Its shortcoming is that the control decisions are generally not the optimal ones, since they are based on local information.

In reality, centralized control is preferred in smaller networks, while localized control is preferred in larger networks. In the latter case, the challenge is to coordinate the control steps taken by each local controller to achieve performances approaching globally optimal ones.

For the traditional hierarchical networks, centralized versions of overload control strategy have been well developed. There is a main controller located at the central call processer which takes control actions in response to all call setup requests. An example is the STATOR method [5].

For networks of distributed architecture, where the role of each processor is equivalent, the situation is much more complex and difficult. Some local control methods have been suggested for this situation [6-8], in which each processor makes decision depending only on its own status and there is no cooperation between each other. Thus they cannot achieve optimal control.

In this paper we propose a centralized control strategy, which achieves globally optimal control through networkwide cooperation. It has the shortcomings of being complex and time-consuming. This leads us to consider modern methods of function approximation. In recent years the use of neural networks for intelligent management and control in telecom networks have been widely studied. For example, Hiramatsu proposed a neural network learning model for call admission control in ATM networks, which found the complex relation between the offered traffic and service quality during stochastic multiplexing [9], Campbell *et al.* proposed a neural network control in capacity allocation for real time implementation [10], Lor and Wong investigated a fast, adaptive and optimal neural network strategy for traffic routing in circuit-switched

networks [11], and so on. In these applications, neural networks can extract the general function from a large number of training examples and generalize it to unknown cases. Hence we propose a neural network control algorithm by using a group of decentralized neural controllers to approximate the complex functions of the centralized controller, thus combining the advantages of both.

The centralized controller serves as the teacher, who generates examples of globally optimal decisions. These examples are used to train the neural controllers off-line, each located on a processor node. After learning, the neural controllers are implemented to infer the control decisions of the teacher based on locally available information.

To evaluate the performance of our method, we perform simulations on a metropolitan network. We compare the behaviors of the proposed local, centralized and neural control methods, referred to as LCM, CCM and NNM respectively. It shows that NNM performs better than LCM both in the throughput and the response to traffic upsurges. Compared with CCM, NNM significantly decreases the computation time for decision making and can be implemented in real-time. So our strategy indeed combines the advantages of both CCM and LCM.

The paper is organized as follows. In section 2, a simplified call processing model is described, and the requirements of an optimal overload control is discussed. We introduce two traditional control methods. One is a local control algorithm (LCM) and the other is a centralized control algorithm (CCM). Their advantages and disadvantages are compared. In section 3, we introduce a radial basis function neural network model and describe its implemention in a telecom network. Simulation results are presented in section 4. The performances of NNM, CCM and LCM under constant heavy traffic and traffic upsurges are compared. We also compare their control errors. In section 5, some general discussions and conclusion are given. Appendix A shows how the processing load of a call processor is calculated, and Appendix B gives the control action of LCM.

2 Overload Control in Telecom Networks

2.1 A Simplified Call Processing Model

Consider a distributed telecom network which consists of N fully connected switch stations (Fig. 1(a)). Call requests between two stations are assumed to arrive as Poisson processes. A call setup process is often complex and may generate various tasks. Here we adopt a simplified model [8], which captures the essential features of real processes: each call setup request initiates five jobs, referred to as jobs 1 to 5 respectively. They represent the jobs of sending dial tones, receiving digits, routing, connecting path and so on. Jobs 1-3 are processed on the originating node, and jobs 4-5 on the terminating node. They consume different service times, denoted by h_i for job i. Time delays between successive jobs are assumed to be stochastic, and obey rectangular distributions, that is, $p(\Delta t) = (\epsilon_2 - \epsilon_1)^{-1}$ for $\epsilon_1 \leq \Delta t \leq \epsilon_2$, and 0 otherwise, where $p(\Delta t)$ is the probability of time delay being Δt, ϵ_1 and ϵ_2 are the minimum and maximum time delays. In this paper we use the parameters shown in Table. 1, where $h_1 = h_3 = h_5 = 50$ ms, $h_2 = 150$ ms and $h_4 = 100$ ms. The time delays between job 1 and jobs 2 & 4 range from 1 to 3 seconds, and those between jobs 2 & 4 and jobs 3 & 5 range from 2 to 8 seconds.

2.2 Objectives of Overload Control

A processor is overloaded if its load status exceeds a predefined threshold. Overload control is implemented by gating new calls. The gate values, i.e. the fraction of admitted calls, are updated periodically. An effective control is to find out the optimal gate values in each period.

The design of an optimal overload control strategy in fully distributed switching systems presents a number of new requirements not encountered in a traditional cen-

Table 1: The Simplified Call Processing Model

Call processing on the originating node	Call processing on the terminating node
job 1 (50 ms)	
$1 \sim 3$ s	
job 2 (150 ms)	job 4 (100 ms)
$2 \sim 8$ s	
job 3 (50 ms)	job 5 (50 ms)

tralized architecture. In this situation, it is important to coordinate the operations of all call processors located on each station. An ideal control algorithm should satisfy the following requirements:

1) Maximum throughput. It is important for a telecom network to maintain high throughput during heavy traffic. Whereas an ideal control should prevent overloading in the network, it should not be done excessively to the extent that the overall throughput is compromised, i.e., overcontrol should be avoided.

2) Balance between stations. This means that all stations share the heavy traffic load. If load balancing is not ensured, the more congested stations will be easily overloaded in the presence of traffic fluctuations.

3) Fairness. The rejecting action to all customers should be fair.

4) Robustness. The control should be robust against changing traffic profiles and partial network breakdown.

5) Easy implementation. The control scheme should be fast, adaptive and simple enough, and can be implemented easily.

Below we introduce two control strateges. Their advantages and disadvantages are compared.

2.3 The Local Control Method (LCM)

Local control methods are the currently adopted overload control strategies in telecommunications networks. Each node monitors its own load and makes decisions independent of all others. As shown in Fig. 1(b), there are two kinds of gate where throttling takes place. The gate values g_i^o and g_i^i denote respectively the acceptance rates of calls outgoing from and incoming to node i. They are updated periodically. Taking into account hardware limitations, control speed and statistical fluctuations, we choose the control period T to be 5 seconds. Priority is given to the incoming calls to maximize the throughput, since they have already consumed processing resources in their originating nodes. When a node is overloaded, the local controller first rejects outgoing call requests. If this is still not effective, the controller further adjusts the incoming gate [7].

For a fair comparison with our proposed methods, we consider a new local control algorithm (LCM), which is better than other local control methods in that the leftover jobs carried forward from the past periods are accommodated. Since a call setup process lasts for about 3-11 seconds (see Table 1), it spans more than one control period, and the control decisions are naturally affected by the presence of the jobs left over from the previous periods.

As derived in Appendix A, the control action at node i during period t should satisfy the capacity constraint given by

$$\sum_j \tau_0 \lambda_{ij}^o(t) g_i^o(t) + \sum_j \hat{\tau}_0 \lambda_{ji}^i(t) g_i^i(t) + \rho_{i-left}(t) \leq \rho_{max}, \tag{1}$$

where the parameters involved are explained below:

(a) τ_0 is the averaged service time for outgoing calls arriving in the current period, $\hat{\tau}_0$ is the corresponding averaged service time for incoming calls. They are different because in the model of Table 1, jobs 1 to 3 contribute to τ_0, whereas jobs 4 to 5 contribute to $\hat{\tau}_0$. They are calculated in Appendix A.

(b) $\rho_{i-left}(t)$ is the leftover load carried forward from the previous periods. It is given by

$$\begin{aligned}\rho_{i-left}(t) &= \tau_1 \sum_j \lambda^o_{ij}(t-1)g^o_i(t-1) + \tau_2 \sum_j \lambda^o_{ij}(t-2)g^o_i(t-2) \\ &+ \hat{\tau}_1 \sum_j \lambda^i_{ji}(t-1)g^i_i(t-1) + \hat{\tau}_2 \sum_j \lambda^i_{ji}(t-2)g^i_i(t-2), \end{aligned} \quad (2)$$

where τ_1 and τ_2 are the averaged service times for outgoing calls having arrived in the previous one and two periods respectively. $\hat{\tau}_1$ and $\hat{\tau}_2$ are the corresponding service times for incoming calls. Again, τ_1 and τ_2 are different from $\hat{\tau}_1$ and $\hat{\tau}_2$, and are derived in Appendix A.

(c) $\lambda^o_{ij}(t)$ is the outgoing call rate from node i to j in period t, and $\lambda^i_{ji}(t)$ is the incoming call rate from node j to i. They are estimated by averaging the call rates over a few control periods. The estimation time should not be too short that the mearsurements are affected by temporal fluctuations, but not too long that the values are insensitive to genuine traffic upsurges. Here we use 5 periods for averaging.

(d) ρ_{max} is the predefined capacity threshold. It is set to 0.85, slightly below the nominal value of 1 to accommodate for traffic fluctuations.

The local controller first maximizes the incoming gate values, and next the outgoing gate values, while satisfying the capacity constraint (1). Explicit expressions are given in Appendix B.

LCM is not an optimal control, for there is no cooperation between different nodes. However, it has the advantages of simplicity and robustness.

2.4 The Optimal Centralized Control Method (CCM)

In the centralized control algorithm, networkwide information is available to the controller. Therefore through cooperative control on each node, only outgoing calls need to be throttled. (Fig. 1(c)). CCM is able to take into account the multiple objectives

prescribed in Section 2.2, in which case the order of priority of the objectives determines the optimization procedure. We consider the maximization of throughput to be the most important, since it is a measure of averaged system performance. Load balancing is next important, since it is a measure of system performance under fluctuations. Fairness comes the third. The technique can be generalized to other choices of priorities. Hence CCM can be implemented as a sequence of linear programming problem. Let the gate value $g_{ij}(t)$ be the acceptance rate for outgoing calls from node i to j in the time period t. They are optimized in the following steps:

Step one: Maximize the throughput $\sum_{(i,j)} \lambda^o_{ij}(t) g_{ij}(t)$ subject to

$$0 \leq g_{ij}(t) \leq 1, \tag{3}$$

$$\tau_0 \sum_j \lambda^o_{ij}(t) g_{ij}(t) + \tau'_0 \sum_j \lambda^o_{ji}(t) g_{ji}(t) + \rho_{i-left}(t) \leq \rho_{max}, \quad 1 \leq i \leq N, \tag{4}$$

where τ_0 has the same meaning as that in LCM, τ'_0 is the corresponding service time for incoming calls. ρ_{i-left} is the leftover load carried forward from the previous periods. It is given by

$$\begin{aligned} \rho_{i-left}(t) &= \tau_1 \sum_j \lambda^o_{ij}(t-1) g_{ij}(t-1) + \tau_2 \sum_j \lambda^o_{ij}(t-2) g_{ij}(t-2) \\ &+ \tau'_1 \sum_j \lambda^o_{ji}(t-1) g_{ji}(t-1) + \tau'_2 \sum_j \lambda^o_{ji}(t-2) g_{ji}(t-2), \end{aligned} \tag{5}$$

where τ_1 and τ_2 have the same meaning as that in LCM. τ'_1 and τ'_2 are the corresponding service times for incoming calls. Note that τ'_0, τ'_1 and τ'_2 are different from $\hat{\tau_0}$, $\hat{\tau_1}$ and $\hat{\tau_2}$ used in LCM, since the former is based on globally available information, whereas the latter is based on the local estimation of a node (see Appendix A).

The above problem can be solved using the active set searching method in linear programming [12]. It turns out that the optimal solution space is often degenerate. Any point in the solution space has the same value of maximum throughput. Removing the degeneracy enables us to optimize the secondary objectives of load balancing and fairness.

Removing the degeneracy is also important for subsequent training of neural networks in NNM. As described in the next section, CCM is used to generate examples for training neural controllers. Degeneracy means the teacher will prescribe different control actions for similar network situations. This is bad for supervised learning since in this case the student will only learn to output the mean value of the teacher's outputs. In order to apply supervised learning to the neural controllers, unambiguous examples should be provided.

Step two: Optimize load balance in the subspace of maximum throughput. At the end of *Step one*, this subspace is defined by a number of equations and inequalities in (3) and (4), referred to as active and inactive constraints respectively. Active and inactive constraints in (4) correspond to full nodes and non-full nodes respectively. We maximize θ in the subspace of maximum throughput, where for each non-full node i,

$$\tau_0 \sum_j \lambda^o_{ij}(t) g_{ij}(t) + \tau'_0 \sum_j \lambda^o_{ji}(t) g_{ji}(t) + \rho_{i-left}(t) + \theta \leq \rho_{max}. \tag{6}$$

Maximizing θ decreases the load of the most congested nodes. As a result, the traffic load is more evenly distributed among the stations. If there is still degeneracy, which is generally the case in our numerical simulation, the third optimization step is needed.

Step three: Optimize fairness by maximizing η in the subspace of maximum throughput and optimal load balance, where

$$\eta \ \leq \ g_{ij}(t) \leq 1, \tag{7}$$

and each $g_{ij}(t)$ denotes an undetermined gate value (inactive constraint) in the previous optimization. Maximizing the lower bound η will avoid unfair rejection in some nodes. In case there is still degeneracy, we apply *Step three* until all degeneracies are lifted.

The method is very time-consuming. On HP 9000 workstations, one turn of decision making for a network of 7 fully connected nodes needs 0.4 seconds. The computational

time grows exponentially with the increase of the size of networks, proportional to N^6, where N is the number of nodes [12]. It is also susceptible to network breakdown and brings heavy load to the signaling network, since networkwide information is necessary.

3 The Neural Network Method (NNM)

A neural network on a processor node receives input about the conditions of the connected call processors, and outputs the corresponding control decisions about the gate values. It acquires this input-output mapping by a learning process using examples generated by CCM. It is difficult to train the neural networks properly using examples generated for a large range of traffic intensity, but on the other hand, training them at a fixed traffic intensity makes them inflexible to changes. Hence for each processor node, we build a group of neural networks, each member being a single layer perceptron trained by CCM using examples generated at a particular background traffic intensity. The final output is an interpolation of the outputs of all members using radial basis functions, which weight the outputs according to the similarity between the background and real-time traffic intensities. This enables the neural controller to make a smooth fit to the desired control function, which is especially important during traffic upsurges.

This network architecture is similar to that of Stokbro *et al* [13], where each hidden unit produces as an output a linear function of the inputs, and the final output is their average weighted by the radial basis functions. Our network differs from theirs in that the outputs of the hidden units are nonlinear sigmoid functions, and that we save the effort of data clustering by taking advantage of the natural clusters according to their background traffic intensities.

3.1 Training a Member of the Group of Neural Networks

For a neural controller associated with a node, the available information includes the measurements, within an updating period of all the outgoing and incoming call attempts, and the processing load of all nodes. Note that the processing load is the only global information fed into the neural controller. These are used to estimate the background load and leftover jobs on itself and other nodes.

To increase the learning efficiency of the neural networks, it is important to preprocess the inputs, so that they are most informative about the teacher control function. From the viewpoint of the neural controller at node i, the constraint of capacity is

$$\sum_j \tau_0 \lambda_{ij}^o g_{ij}(t) + \sum_j \hat{\tau}_0 \lambda_{ji}^i(t) + \rho_{i-left}(t) \leq \rho_{max}. \tag{8}$$

At the same time, the controller at node i should consider the constraints of capacity at other nodes $j \neq i$, that is

$$\tau_0' \lambda_{ij}^o g_{ij}(t) + \tau_0 \lambda_{ji}^o g_{ji}(t) + \rho_{ij-left}(t) + \rho_{j-back}(t) \leq \rho_{max}, \qquad j \neq i. \tag{9}$$

where the first two terms are the processing load on node j generated by the traffic flow between node i and j, and $\rho_{ij-left}(t)$ is the corresponding leftover load. $\rho_{j-back}(t)$ is the background processing load between node j and other nodes excluding node i. To the neural controller, the information of λ_{ji}^o and g_{ji} for $j \neq i$ is not available. To assess the processing load on node j, it has to estimate the traffic flow $\lambda_{ji}^o g_{ji}$ (measured by the arrival rate of admitted job 1) from the knowledge of λ_{ji}^i (measured by the arrival rate of job 4). We estimate $\lambda_{ji}^o g_{ji}(t)$ to be $\lambda_{ji}^i(t)$. $\rho_{ij-left}(t)$ is given, in analogy to (5), by

$$\begin{aligned} \rho_{ij-left}(t) &= \tau_1 \lambda_{ij}^o(t-1) g_{ij}(t-1) + \tau_2 \lambda_{ij}^o(t-2) g_{ij}(t-2) \\ &+ \tau_1' \lambda_{ji}^o(t-1) g_{ji}(t-1) + \tau_2' \lambda_{ji}^o(t-2) g_{ji}(t-2). \end{aligned} \tag{10}$$

$\rho_{j-back}(t)$ is estimated by averaging over a few periods.

For simplicity, we rewrite the equations (8) and (9) as

$$\sum_j \tau_0 \lambda^o_{ij} g_{ij}(t) \leq \tilde{\rho}_i, \tag{11}$$

$$\tau'_0 \lambda^o_{ij} g_{ij}(t) \leq \tilde{\rho}_j, \quad j \neq i. \tag{12}$$

where $\tilde{\rho}_i = \rho_{max} - \sum_j \hat{\tau}_1 \lambda^i_{ji}(t) - \rho_{i-left}(t)$ and $\tilde{\rho}_j = \rho_{max} - \tau_0 \lambda^o_{ji} g_{ji}(t) - \rho_{ij-left}(t) - \rho_{j-back}(t)$.

To find the most informative inputs to the neural networks, we consider for illustration a simple network of 3-fully connected nodes. The feasible solution space satisfying the above constraints is shaded in Fig. 2. The following variables are important in reflecting the geometry of the shaded region: (a) the range along the direction of $g_{ij}(t)$, given by $\tilde{\rho}_j / \tau'_0 \lambda_{ij}$ for $j \neq i$. Since $g_{ij}(t)$ lies between 0 and 1, we let min $(\tilde{\rho}_j / \tau'_0 \lambda_{ij}$, 1) to be $N-1$ inputs to the neural control at node i. (b) the distance of the plane corresponding to constraint (11) from the origin, given by $\tilde{\rho}_i / \tau_0 [\sum_j (\lambda^o_{ij}(t))^2]^{1/2}$. Since this is bounded above by $\sqrt{N}$, we let min $(\tilde{\rho}_i / \tau_0 [\sum_j (\lambda^o_{ij}(t))^2]^{1/2}$, $\sqrt{N})$ to be the N^{th} input to the neural network at node i.

The other $N-1$ inputs consist of the outgoing call attempts λ^o_{ij} for node i. We normalize λ^o_{ij} by a factor $\sqrt{\sum_l (\lambda^o_{il})^2}$, since according to the constraints in CCM, they represent the optimization direction in the space of gate values.

The above inputs form a $2N-1$ dimensional vector ξ^1 fed to each neural network in the group, each trained by a distinct training set of examples. The k^{th} member outputs the gate values g^k_{ij} according to

$$g^k_{ij} = f(\sum_{n=1}^{2N-1} J^k_{ijn} \xi^1_n + J^k_{ij0}), \tag{13}$$

where $f(h) = (1 + e^{-h})^{-1}$ is the sigmoid function. The couplings J^k_{ijn} and the bias J^k_{ij0} are obtained during the learning process by gradient descent minimization of an energy function

$$E = \frac{1}{2} \sum_{k,\mu} (O^{k,\mu}_{ij} - g^{k,\mu}_{ij})^2 \tag{14}$$

where $O_{ij}^{k,\mu}$ is the optimal decision of g_{ij} prescribed by the teacher for example μ in the k^{th} training set, and $g_{ij}^{k,\mu}$ is the output of the k^{th} member of the group of neural networks.

3.2 Implementation of the Group of Neural Networks

Consider the part of neural controller for calculating the gate value g_{ij}, as shown in Fig. 3 (the other parts have the same structure). The k^{th} hidden unit is trained at a particular traffic intensity, and outputs the decision $g_{ij}^k(\xi^1)$ for the $2N-1$ dimensional input vector ξ^1 discribed in Section 3.1.

To weight the contribution of the k^{th} output, we consider a $N-1$ dimensional input vector ξ^2 which consists of the call rates $\lambda_{ij}^0(t)$, $j \neq i$. The weight $f^k(\xi^2)$ is the radial-basis-function (RBF) [14] given by

$$f^k(\xi^2) = \frac{\exp[-(\xi^2 - \mu^k)^2/2\sigma_k^2]}{\sum_l \exp[-(\xi^2 - \mu^l)^2/2\sigma_l^2]} \tag{15}$$

where μ^k is the k^{th} RBF center, and σ_k is the size of the RBF cluster. In our case, μ^k is the input vector ξ^2 averaged over the k^{th} training set of examples, and describes the backgound traffic intensity. σ_k^2 is chosen to be the variance of the Poisson traffic at the k^{th} RBF center, i.e. $\sum_{j\neq i}\langle\lambda_{ij}^o\rangle/T$. This is slightly different from the usual choice of σ_k^2 being the variance of the k^{th} training set, which is smaller in our simulations. In fact, it turns out that our choice yields a better performance in simulations.

The final output of the neural network is a combination of the weighted outputs of all hidden units, that is,

$$g_{ij}(\xi^1, \xi^2) = \sum_k f^k(\xi^2) g_{ij}^k(\xi^1). \tag{16}$$

Since the numerator of (15) is a decreasing function of the distance between the vector ξ^2 and μ^k, the RBF center nearest to ξ^2 has the largest weight. If ξ^2 moves

between the RBF centers, their relative weights change continuously, hence providing a smooth interpolation of the control function.

4 Simulation results

To compare the above three methods, we perform simulations on part of the Hong Kong metropolitan network (for convenience we call it the Jumbo Network), which consists of 7-fully connected switch stations (see Fig1.(a)). The call arrival rates between different nodes under normal traffic condition are shown in Table 2. Call attempts are generated according to the Poisson process, and accepted with probabilities given by the corresponding gate values. The accepted calls will queue in the buffer waiting for service. If the queueing time is too long, customers may lose patience and abandon the call attempts. This is the overflow process [15]. When the traffic is well controlled, the chance of overflow is small. It usually happens during traffic upsurges where a large amount of calls arrive simultaneously. In our simulation, we assume that the overflow process is stochastic and satisfies an exponential distribution. The survival probability p of a call after waiting for t seconds is assumed to be $p = \min\{1, e^{-0.35(t-1)}\}$. Thus within 1 second, there is no abandonment. After 5 seconds of waiting, the probability the customer still in the queue is around 25%.

The RBF centers of the neural networks are chosen as 1, 2, 3, 4, 6 and 8 multiples of the normal traffic intensity. To generate examples for neural network training by CCM algorithm, we simulate the traffic corresponding to each RBF center for more than 2×10^5 seconds. The data of network scenarios and their associated globally optimal decisions are collected to train the neural controllers off-line. Fig. 4 gives an example of neural control during constant heavy traffic. The controlled load of each processor fluctuates around the predefined threshold value 0.85. The performance of the three control methods are compared in the following aspects:

4.1 Steady Throughput

Fig. 5 compares the throughputs of the network under different steady-state traffic intensities. The simulation for each case are done for 4000 seconds. We see that the neural control performs comparably with the centralized teacher, and has a large improvement in throughput over the local control for a large range of traffic intensities.

4.2 Control Error

In reality control errors are unavoidable due to statistical fluctuations. They may be due to the stochasticity of call arrivals, the uncertainty in measuring the traffic rates, the stochasticity of time delays in job arrivals etc. We define the control error (CE) as the fraction of the processing load which exceeds the nominal value 1, given by

$$CE = \frac{\sum_{i,t}(\rho_i(t) - 1)\theta(\rho_i(t) - 1)}{\sum_{i,t} \rho_i(t)} \tag{17}$$

where $\rho_i(t)$ is the actual load on node i in the control period t. $\theta(x)$ is the step function, which equals 1 when $x > 0$ and 0 otherwise. CE reflects the stability of control to the fluctuations. Fig.6 compares the control error of the three methods during constant traffic. It shows that CCM has lowest error at all traffic intensities. For light traffic, NNM and LCM have comparable control errors, whereas for heavy traffic, NNM performs better than LCM.

4.3 Traffic Upsurges

Of particular interest to network management is the response of the system to traffic upsurges. In reality this occurs in such cases as phone-in programs, telebeting and the hoisting of typhoon signals, when the amount of call attempts abruptly increases. It is expected that control schemes should respond as fast as possible to accommodate the changing traffic condition.

We model two cases of traffic upsurges. The first case is a traffic upsurge in all nodes. Fig. 7(a) shows how the system responds when the normal traffic intensity becomes sixfold at $t = 40s$. We also measured the averaged control errors of three methods for the subsequent $50s$. We see that NNM has a throughput higher than CCM, but with a slight, tolerable compromise in control error. They are both much better than LCM.

The second is that traffic intensites only increase in part of the network. Responses to an upsurge of incoming and outgoing traffic of node 1 are shown in Fig. 7(b), leading to the same conclusion.

Neural controller significantly decreases the time for making decisions. For the network we simulated, it is about 10% of the CPU time of CCM. Hence NNM can be implemented in real time.

5 Conclusion and Discussions

In summary, we have found a neural network algorithm for overload control in telecommunication systems. The neural controllers are implemented in each station and learns the controlling functions prescribed by an optimal centralized teacher. Simulations show that NNM performs better than the local controller in both the throughput and the response to traffic upsurges. Compared with the centralized teacher, the neural controller performs comparably in the response to traffic upsurges. It is interesting to note that NNM significantly decreases the time for making decisions, and hence can be implemented in real time. NNM combines the advantages of both LCM and CCM, and achieves a simple, adaptive, robust and near-optimal control.

Our method is more powerful if the network traffic is more inhomogeneous. To see this, we simulate a network with an extremely inhomogeneous traffic profile. The normal traffic rates are shown in Table 3. Again, we choose RBF centers of neural

networks as 1, 2, 3, 4, 6 and 8 times the normal traffic intensity and repeat the above procedure of neural training. Fig. 8 shows the steady throughput of the network at different traffic intensities under the control of the three methods. We see that NNM has a significant improvement in throughput over LCM, which is much larger than that in Fig. 5. In the case of completely homogeneous traffic, our simulation shows that there is no difference between the three methods in steady throughput, and only a small difference in response to traffic upsurges. Since traffic is often inhomogeneous in reality, NNM is a realistic control method.

For a control method to work outside the simulator, it should be robust against the choice of the call processing model. In our approach, the estimation of the processing load is based on a simplified call processing model, by which we calculate the averaged service times. However, it may not be easy in practice to estimate the distribution of service times. For example, the time delay between different jobs may not be a rectangular distribution. We perform simulations on a modified model of call processing, in which the time delays between different jobs obey a Gaussian distribution (truncated when the argument is negative), whereas the controllers operate by assuming the model of rectangular distributions with the same averages as shown in Table 1, i.e. the controllers make an unprecise assumption. Simulations show that there is no change in the network throughput and only a little increase in control errors.

In our approach we choose normal traffic rates and its multiples as RBF centers of neural networks, without doing any data clustering which is generally needed in neural network training, and find that it works well. In practice, normal traffic rates are available by taking statistics for sufficiently long time in the normal traffic condition. For networkwide increase in traffic, multiples of normal traffic rates can indeed capture the features of traffic profiles and can be chosen as RBF centers. However, it is sometimes difficult to collect data for high traffic situations. To study the effects of the

uneven distribution of training examples, we consider a time dependent traffic pattern as shown in Fig. 9, in which the training examples for high traffic are increasingly rare. It turns out that the performance of the resultant neural controllers is approximately the same as those trained at multiples of the normal traffic described in section 4.

Some remarks about the methodology of the teacher and student controllers are relevant here. In our problem, the teacher is a complex optimization task with multiple objectives. This is realized through stepwise optimizations, which is a process of lifting degeneracies with performance criteria optimized in order of priority. In our case the order is network throughput, balance between stations, and fairness. This approach is in contrast to the more conventional one-step optimization strategy, in which all performance criteria are considered simultaneously, but with relative weights tuned according to priority. In fact, the stepwise optimization process is equivalent to the one-step approach if we take the ratios of the relative weights of successively important criteria to approach infinity. This saves the effort in choosing the reliable weights in the one-step optimization. Furthermore, the computational time for the case of many variables is reduced, and the linearized program does not suffer from convergence problems.

Finally, we remark on the implications of our work to general issues of distributed control. Instead of learning the teacher task by a sophisticated student network, the task is divided among a group of local student networks with simple architecture, which cooperate in the control function. This methodology successfully avoids the shortcomings of traditional centralized and local control methods, and combines their advantages. The control technique can be generalized to the distributed control of many large systems such as the ATM network and the wireless cellular network.

Acknowledgements

This work was supported by the Hong Kong Telecom Institute of Information Technology, HKUST. We would like to thank Prof. Cao Xiren for useful suggestions, and Hong Kong Telecom for providing data of part of the Hong Kong metropolitan network.

Appendix A: The Estimation of Processing Load

For a node in the network, the outgoing and incoming call rates of the node are measured by averaging the number of arriving job 1 and job 4 respectively. Suppose $n^o_{ij}(t)$ is the number of outgoing calls from node i to j in t^{th} control period of duration T, and $n^i_{ji}(t)$ the number of incoming calls from j to i. The outgoing and incoming traffic rates in the coming period $t+1$ are estimated by averaging over the past L periods, that is,

$$\lambda^o_{ij}(t+1) = \sum_{l=0}^{L-1} n^o_{ij}(t-l)/LT, \tag{18}$$

$$\lambda^i_{ji}(t+1) = \sum_{l=0}^{L-1} n^i_{ji}(t-l)/LT. \tag{19}$$

Here we use $T = 5$ s and $L = 5$.

For a centralized controller, the outgoing call rates of all nodes are available; whereas for a local controller associated with a node, the only available local information are the outgoing call rates from this node and the incoming call rates to it. Hence they estimate the processing load in different ways.

1. Estimates for Centralized Control

Consider setting up a call from node i to j. As shown in Fig. 10(a), let Δt_1 be the call arrival time in the current period t, Δt_2 and Δt_3 be the time delays from job 1 to 2 (also job 1 to 4) and job 2 to 3 (also job 4 to 5) respectively.

Let τ_0, τ_1 and τ_2 be the averaged service times for an outgoing call on the originating

node in the period t, and the future periods $t+1$ and $t+2$ respectively. Then

$$\begin{aligned}
\tau_0 &= h_1 + \frac{h_2}{T}\int_0^T d\Delta t_1 \int d\Delta t_2 P(\Delta t_2)|_{0\le \Delta t_1+\Delta t_2\le T} \\
&\quad + \frac{h_3}{T}\int_0^T d\Delta t_1 \int d\Delta t_2 P(\Delta t_2)\int d\Delta t_3 P(\Delta t_3)\,|_{0\le \Delta t_1+\Delta t_2+\Delta t_3\le T} \qquad (20)\\
\tau_1 &= \frac{h_2}{T}\int_0^T d\Delta t_1 \int d\Delta t_2 P(\Delta t_2)\,|_{T\le \Delta t_1+\Delta t_2\le 2T} \\
&\quad + \frac{h_3}{T}\int_0^T d\Delta t_1 \int d\Delta t_2 P(\Delta t_2)\int d\Delta t_3 P(\Delta t_3)\,|_{T\le \Delta t_1+\Delta t_2+\Delta t_3\le 2T} \qquad (21)\\
\tau_2 &= \frac{h_2}{T}\int_0^T d\Delta t_1 \int d\Delta t_2 P(\Delta t_2)\,|_{2T\le \Delta t_1+\Delta t_2\le 3T} \\
&\quad + \frac{h_3}{T}\int_0^T d\Delta t_1 \int d\Delta t_2 P(\Delta t_2)\int d\Delta t_3 P(\Delta t_3)\,|_{2T\le \Delta t_1+\Delta t_2+\Delta t_3\le 3T} \qquad (22)
\end{aligned}$$

Let τ_0', τ_1' and τ_2' be the averaged service times on the terminating node in the current period t, and the future periods $t+1$ and $t+2$ respectively. Then

$$\begin{aligned}
\tau_0' &= \frac{h_4}{T}\int_0^T d\Delta t_1 \int d\Delta t_2 P(\Delta t_2)\,|_{0\le \Delta t_1+\Delta t_2\le T} \\
&\quad + \frac{h_5}{T}\int_0^T d\Delta t_1 \int d\Delta t_2 P(\Delta t_2)\int d\Delta t_3 P(\Delta t_3)\,|_{0\le \Delta t_1+\Delta t_2+\Delta t_3\le T} \qquad (23)\\
\tau_1' &= \frac{h_4}{T}\int_0^T d\Delta t_1 \int d\Delta t_2 P(\Delta t_2)\,|_{T\le \Delta t_1+\Delta t_2\le 2T} \\
&\quad + \frac{h_5}{T}\int_0^T d\Delta t_1 \int d\Delta t_2 P(\Delta t_2)\int d\Delta t_3 P(\Delta t_3)\,|_{T\le \Delta t_1+\Delta t_2+\Delta t_3\le 2T} \qquad (24)\\
\tau_2' &= \frac{h_4}{T}\int_0^T d\Delta t_1 \int d\Delta t_2 P(\Delta t_2)\,|_{2T\le \Delta t_1+\Delta t_2\le 3T} \\
&\quad + \frac{h_5}{T}\int_0^T d\Delta t_1 \int d\Delta t_2 P(\Delta t_2)\int d\Delta t_3 P(\Delta t_3)\,|_{2T\le \Delta t_1+\Delta t_2+\Delta t_3\le 3T} \qquad (25)
\end{aligned}$$

For the call processing model in Table 1, $\tau_0 = 141$ ms, $\tau_1 = 88$ ms, $\tau_2 = 21$ ms, $\tau_0' = 61$ ms, $\tau_1' = 68$ ms and $\tau_2' = 21$ ms.

Based on the knowledge of λ_{ij}^0 for all nodes, the centralized controller estimates the processing load of of node i in the period t as in the left hand side of (4), where $\rho_{i-left}(t)$ is the leftover load carried forward from the previous periods given in (5).

2. Estimates for Local Control

For a local controller associated with a node i, the information of λ_{ji}^o for $j \neq i$ is not available, and it has to estimate the processing load from the only available information of λ_{ji}^i for $j \neq i$. As shown in Fig. 10(b), let $\hat{\tau}_0$, $\hat{\tau}_1$ and $\hat{\tau}_2$ be the averaged service times

for an incoming call in the current period t, and the future periods $t+1$ and $t+2$ respectively. Again, assuming $\Delta t_1'$ is uniformly distributed,

$$\hat{\tau}_0 = h_4 + \frac{h_5}{T}\int_0^T d\Delta t_1' \int d\Delta t_2' P(\Delta t_2') \mid_{0 \le \Delta t_1' + \Delta t_2' \le T} \quad (26)$$

$$\hat{\tau}_1 = \frac{h_5}{T}\int_0^T d\Delta t_1' \int d\Delta t_2' P(\Delta t_2') \mid_{T \le \Delta t_1' + \Delta t_2' \le 2T} \quad (27)$$

$$\hat{\tau}_2 = \frac{h_5}{T}\int_0^T d\Delta t_1' \int d\Delta t_2' P(\Delta t_2') \mid_{2T \le \Delta t_1' + \Delta t_2' \le 3T} \quad (28)$$

For the call processor model in Table 1, $\hat{\tau}_0 = 107.5$ ms, $\hat{\tau}_1 = 35$ ms and $\hat{\tau}_2 = 7.5$ ms. So the local controller at node i estimates the processing load in the period t as in left hand side of (1), and the leftover load $\rho_{i-left}(t)$ as in (2).

Appendix B: The Control Action of LCM

The local controllers in LCM adjust the gate values in the following way:

(a)*high load*:

if $\rho_{max} \le \rho_{i-left}(t)$,

$g_i^i = 0$, $g_i^o = 0$.

(b)*intermediate load*:

1. *if* $\rho_{i-left} \le \rho_{max} \le \rho_{i-left}(t) + \hat{\tau}_0 \sum_j \lambda_{ji}^i(t)$,

$g_i^i = [\rho_{max} - \rho_{i-left}(t)]/\hat{\tau}_0 \sum_j \lambda_{ji}^i(t)$, $g_i^o = 0$.

2. *if* $\rho_{i-left}(t) + \sum_j \hat{\tau}_0 \lambda_{ji}^i(t) \le \rho_{max} \le \rho_{i-left}(t) + \sum_j \hat{\tau}_0 \lambda_{ji}^i(t) + \sum_j \tau_0 \lambda_{ij}^o(t)$.

$g_i^i(t) = 1$, $g_i^o = [\rho_{max} - \rho_{i-left}(t) - \sum_j \hat{\tau}_0 \lambda_{ji}^i(t)]/\sum_j \tau_0 \lambda_{ij}^o(t)$.

(c) *light load*:

if $\rho_{max} \ge \rho_{i-left}(t) + \sum_j \hat{\tau}_0 \lambda_{ji}^i(t) + \sum_j \tau_0 \lambda_{ij}^o(t)$,

$g_i^i = 1$, $g_i^o = 1$.

References

[1] A. E. Eckberg and P. E. Wirth, "Switch Overload and Flow Control Strategies in ISDN Enviroment", *Traffic Engineering for ISDN Design and Planning*, pp. 425-434, 1988.

[2] B. Sanso, F.Soumis and M. Gendreau, "Centralized and Decentralized Stochastic Routing Models in Telecommunication Networks", *Telecommunication Systems - Modeling, Analysis, Design and Management*, vol. 1, no. 2, pp. 133-148, 1993.

[3] G. M. Woodruff, R. G. H. Rogers and P. S. Richards, "A Congestion Control Framework for High-speed Integrated Packetized Transport", *Proc. GLOBECOM'88*, pp. 7.1.1-7.1.5, 1988.

[4] M. Schwartz, "Network Management and Control Issues in Multimedia Wireless Networks", *IEEE Personal Communications*, 1995.

[5] P. Hanselka, J. Oehlerich and G. Wegmann, "Adaptation of the Overload Regulation Method STATOR to Multiprocessor Controls and Simulation Results", *ITC-12*, pp. 395-401, 1989.

[6] D. Manfield, B. Denis, K. Basu and G. Rouleau, "Overload Control in a Hierarchical Switching System", *ITC-11*, pp. 894-900, 1985.

[7] M. Villen-Altamirano, G. Morales-Andres and L. Bermejo-Saez, "An Overload Control Strategy for Distributed Control Systems", *ITC-11*, pp. 835-841, 1985.

[8] J. S. Kaufman and A. Kumar, "Traffic Overload Control in a Fully Distributed Switching Environment", *ITC-12*, pp. 386-394,1989.

[9] A. Hiramatsu, "ATM Communications Network Control by Neural Networks", *IEEE Transactions on Neural Networks*, vol. 1, pp. 122-130, 1991.

[10] P. K. Campbell, M. dale, H. L. Ferra and A. Kowalczyk, "Experiments wth Neural Networks for Real Time Implementation of Control", *Advances in Neural Information Processing System 8*, Cambridge, MA: MIT Press, 1995.

[11] W. K. F. Lor and K. Y. M. Wong, "Decentralized Neural Dynamic Routing in Circuit-switched Networks", *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications 2* (IWANNT-95), Ed. J. Alspector et al., Lawrence Erlbaum Associates, New Jersy, pp. 137-144.

[12] M. J. Best and K. Ritter, *Linear Programming: Active Set Analysis and Computer Programs*, 1985, Englewood Cliffs, NJ. : Prentice-Hall.

[13] K. Stokbro, D. K. Umberger and J. A. Hertz, "Exploiting Neurons with Localized Receptive Fields to Learn Chaos", *Complex Systems* **4**, pp. 603-622, 1990.

[14] J. A. Hertz, A. Krogh and R. G. Palmer, *Introduction to the Theory of Neural Computation*, London, Addison-Wesley, 1991.

[15] A. W. Berger, "Comparison of Call Gapping and Percent Blocking for Overload Control in Distributed Switching Systems and Telecommunications Networks", *IEEE Transactions on Communications*, vol. 39, pp. 574-580, 1991.

Figure Captions

Fig.1:(a) A 7-Node fully connected network of switch stations; (b) Local control method; (c) Centralized control method.

Fig.2: A simple diagram to illustrate the solution space of inequalities (10) and (11) for node 1.

Fig.3: The part of neural network for calculating gate value g_{ij}.

Fig.4: The processing load of node 1 in Jumbo Network under the control of NNM. The traffic intensity is 6 times the normal case.

Fig.5: Network throughput under constant traffic. The traffic intensities are measured in multiples of the normal rates.

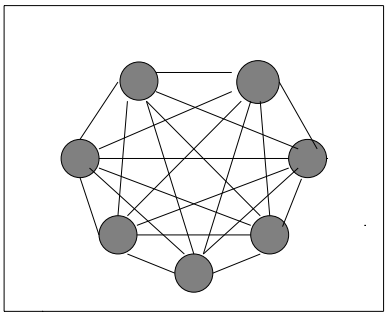
Fig.6: Control errors under different traffic intensities.

Fig7: (a) Network throughput during a traffic upsurge on all nodes. The traffic intensities of all nodes increase to 6 times at $t = 40s$. CEL, CEC and CEN are the averged control errors of LCM, CCM and NNM respectively within $50s$ after traffic increase. (b) Network throughput during a traffic upsurge at node 1. The traffic intensity of node 1 increases to 8 times at $t = 40s$.

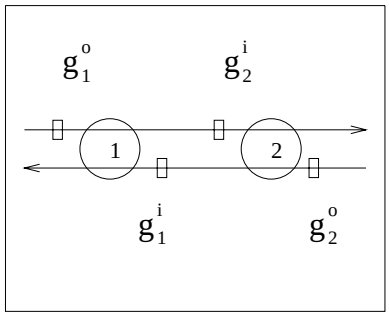
Fig.8: Network throughput under constant traffic. The traffic intensities are measured in multiples of the normal rates shown in Table 3.

Fig.9: A time dependent traffic pattern.

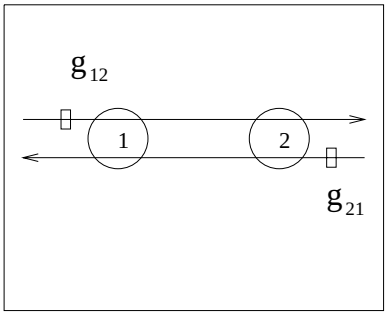
Fig.10: The time structure of call setup processes. (a) A outgoing call setup process; (b) A incoming call setup process.



(a)



(b)



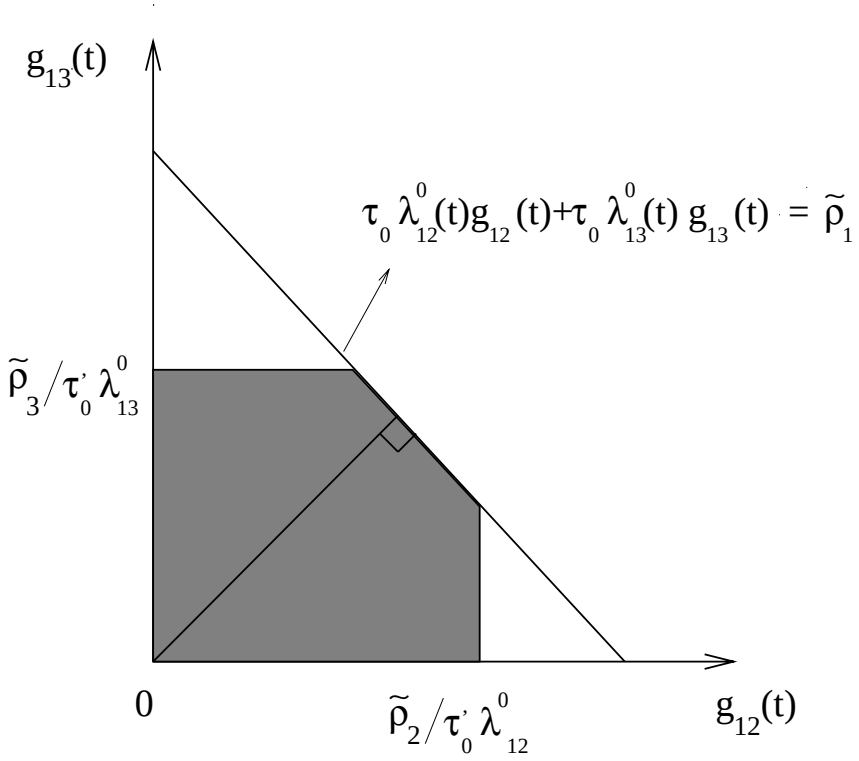
(c)

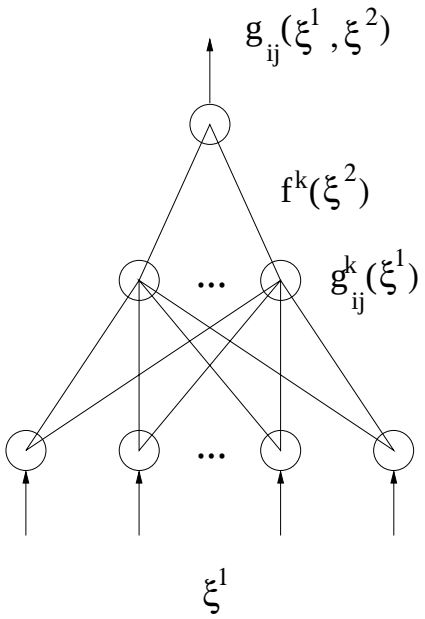
	S_0	S_1	S_2	S_3	S_4	S_5	S_6
S_0	0	480	1070	1040	1640	280	670
S_1	360	0	220	320	390	240	300
S_2	900	400	0	2100	1550	450	520
S_3	700	410	2090	0	1020	270	410
S_4	1080	280	1300	970	0	380	400
S_5	250	220	290	170	230	0	210
S_6	500	260	490	430	450	230	0

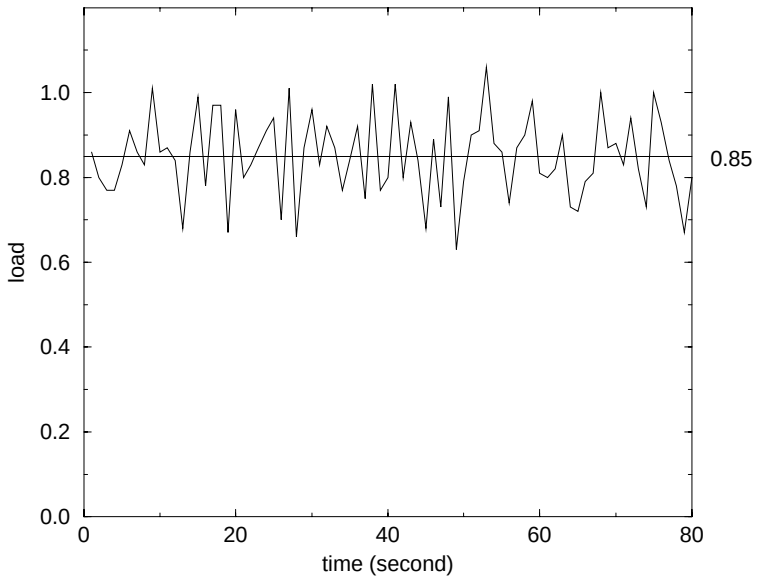
Table 2: Call arrival rates (per hour) of the Jumbo Network in the condition of normal traffic.

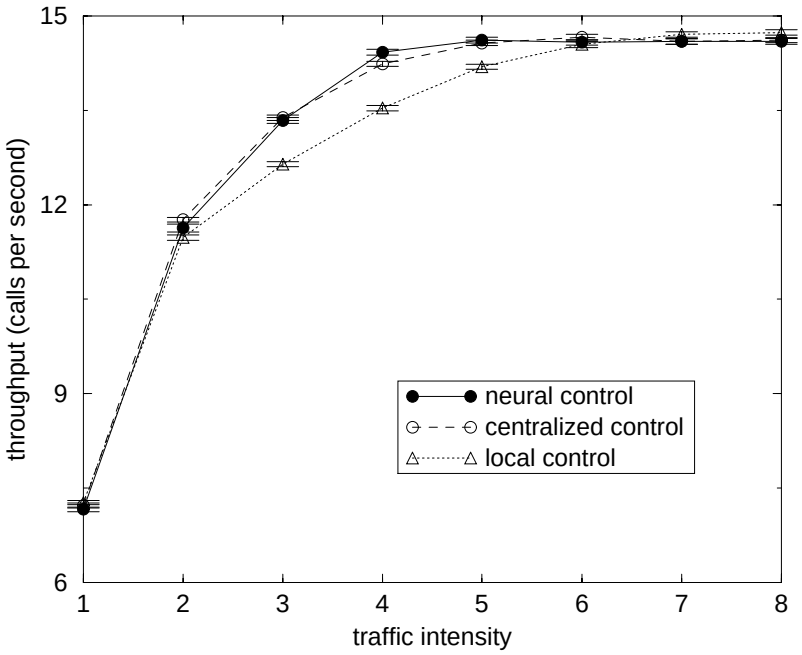
	S_0	S_1	S_2	S_3	S_4	S_5	S_6
S_0	0	180	180	180	180	180	180
S_1	1800	0	1800	1800	1800	1800	1800
S_2	180	180	0	180	180	180	180
S_3	1800	1800	1800	0	1800	1800	1800
S_4	180	180	180	180	0	180	180
S_5	1800	1800	1800	1800	1800	0	1800
S_6	180	180	180	180	180	180	0

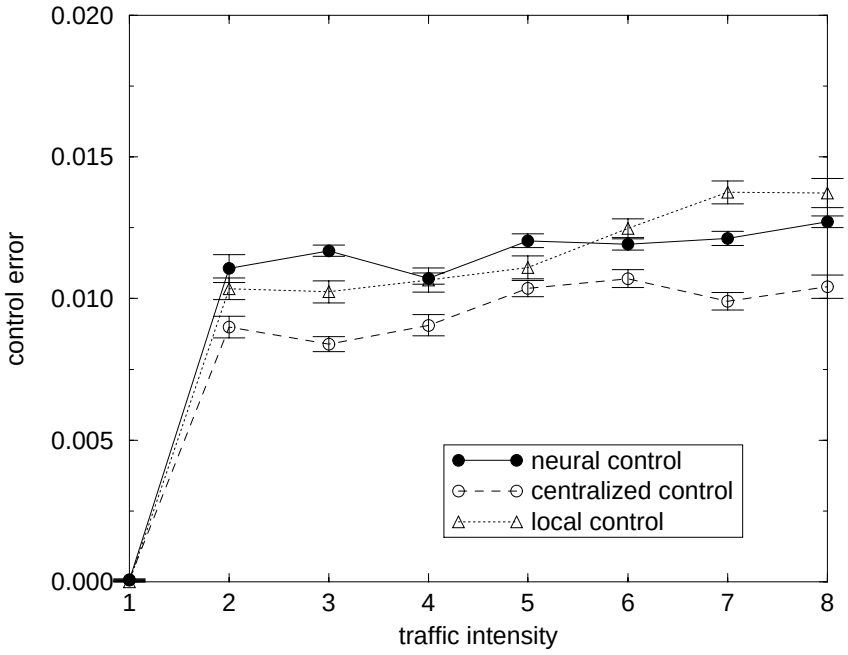
Table 3: Call arrival rates (per hour) of an artificial network with extremely asymmetry traffic.

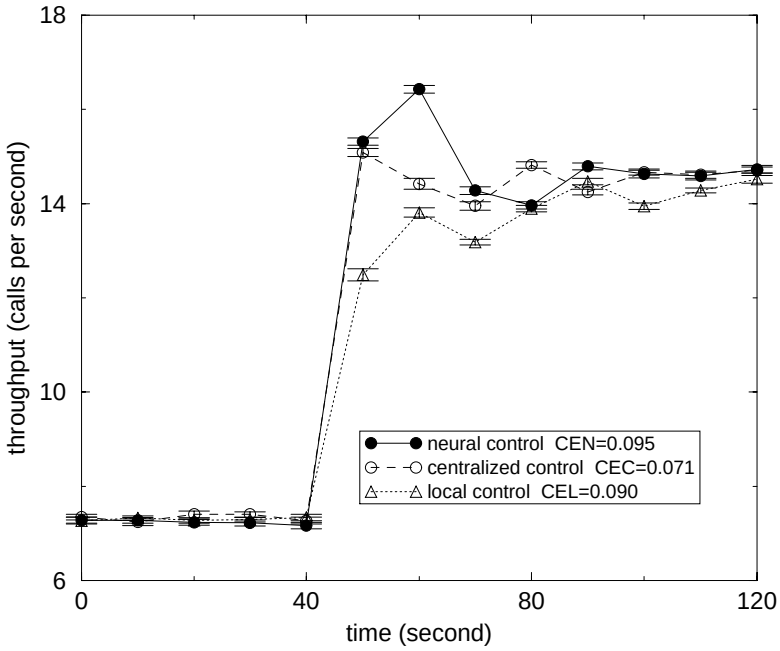



$g_{ij}(\xi^1, \xi^2)$
$f^k(\xi^2)$
$g^k_{ij}(\xi^1)$
ξ^1

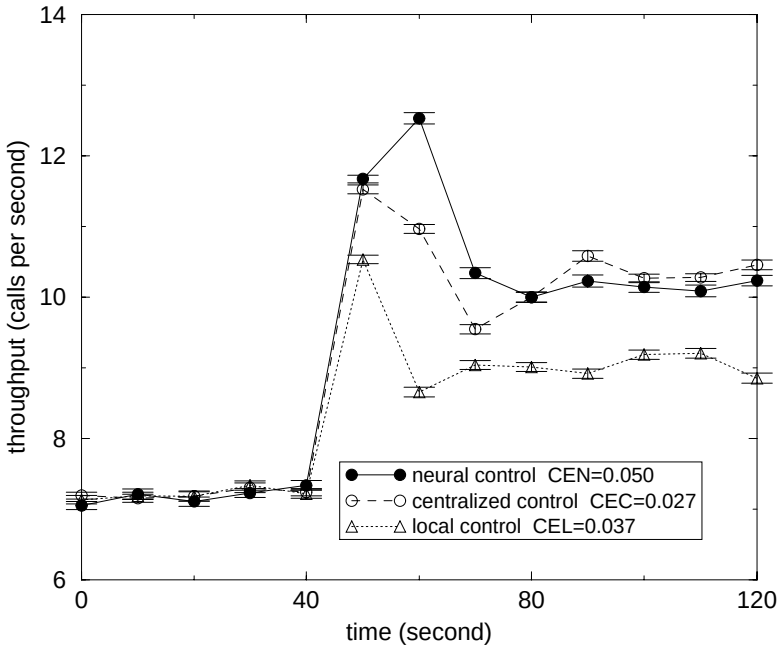

load
time (second)
0.85
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80


15
12
9
6
1
2
3
4
5
6
7
8
throughput (calls per second)
traffic intensity
neural control
centralized control
local control

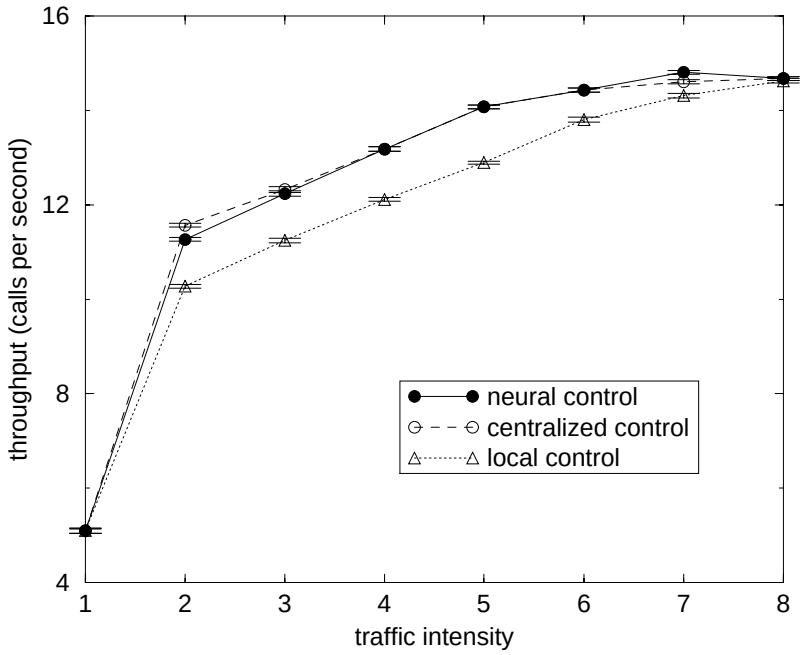

0.020
0.015
0.010
0.005
0.000
1
2
3
4
5
6
7
8
control error
traffic intensity
neural control
centralized control
local control

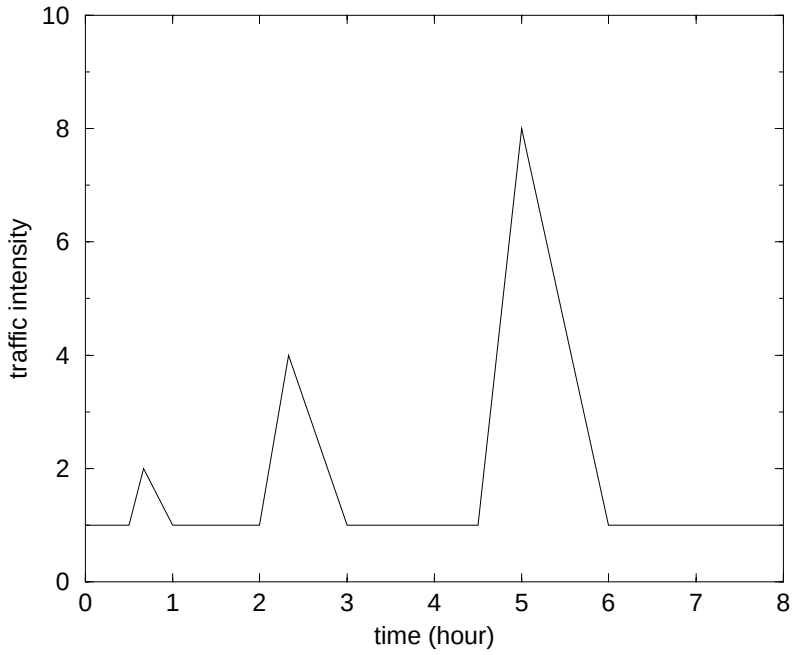

18
14
10
6
0
40
80
120
throughput (calls per second)
time (second)
neural control CEN=0.095
centralized control CEC=0.071
local control CEL=0.090

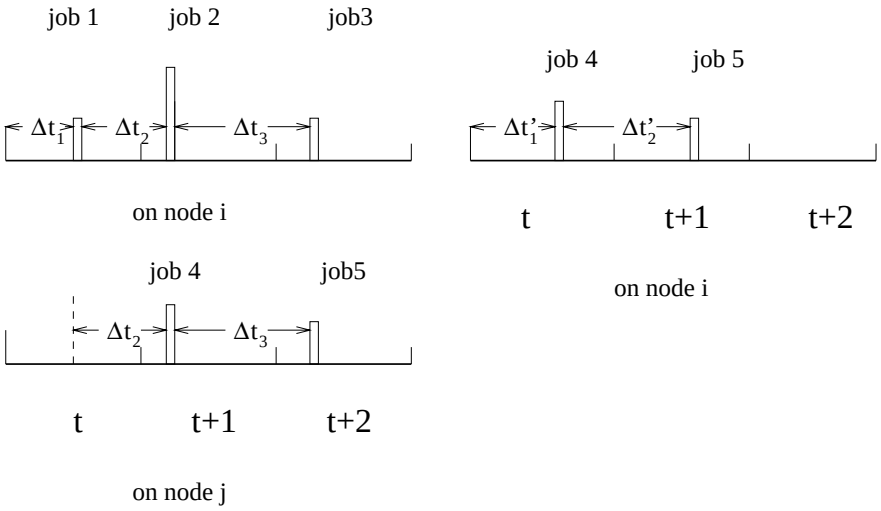
(a)


14
12
10
8
6
0
40
80
120
throughput (calls per second)
time (second)
neural control CEN=0.050
centralized control CEC=0.027
local control CEL=0.037

(b)


throughput (calls per second)
traffic intensity
neural control
centralized control
local control
16
12
8
4
1
2
3
4
5
6
7
8


traffic intensity
time (hour)
10
8
6
4
2
0
0
1
2
3
4
5
6
7
8


job 1
job 2
job3
Δt1
Δt2
Δt3
on node i
job 4
job5
Δt2
Δt3
t
t+1
t+2
on node j
job 4
job 5
Δt'1
Δt'2
t
t+1
t+2
on node i

(a) (b)