2007 International Symposium on Nonlinear Theory and its Applications
NOLTA'07, Vancouver, Canada, September 16-19, 2007

An efficient routing strategy with load-balancing for complex networks

Takayuki Kimura and Tohru Ikeguchi

Graduate School of Science and Engineering, Saitama University
255 Shimo-Ohkubo, Sakura-ku, Saitama 338-8570, Japan
Email: kimura@nls.ics.saitama-u.ac.jp

Abstract—
Congestion of packets in the Internet is the most undesirable problem to securely communicate between end users. Thus, many approaches have been attempting to resolve such a problem. We have also proposed a routing strategy with chaotic neurodynamics[1]. The chaotic neurodynamics is introduced to alleviate the packet congestion, then, the routing strategy shows high performance for complex networks comparing with the shortest path approach (the Dijkstra algorithm). In the routing strategy, an adjacent node from which a path to a destination of a packet is the shortest is determined as a transmitting node. However, if we obtain more information, we expect that the performance of the routing strategy becomes higher. From this view point, in this paper, we extend the routing strategy, combining information of the shortest path and waiting times at adjacent nodes. We show that the improved routing strategy has high performance for the complex networks.

1. Introduction

Packet congestion is the most undesirable issue in the Internet because it leads to loss and delay of packets. The packet congestion is related to a spatial structure of a computer network and a routing strategy. To alleviate the packet congestion of the computer networks, there are two directions of research.

The first one is to analyze a topology of a computer network in detail, then, propose a proper routing strategy which works well on the computer network. For example, Arenas et al. focused on Clay-tree networks[2]. Scale-free networks proposed by Barábasi and Albert[3] have also been widely studied[4, 5]. The second one is to propose a routing strategy which works well on various topologies of the computer networks. Horiguchi et al. proposed a routing strategy based on mutual connection neural networks[6]. In addition, to improve the performance of the routing strategy[6], they introduced the reinforcement learning[7], then, they showed the routing strategy has high performance for a fractal lattice, a small-world networks[8] and the scale-free networks[3]. Kimura et al. also modified the routing strategy[6] in which a stochastic effect is introduced[9], and the modified routing strategy showed high performance for small-world type randomized networks and the scale-free networks[3]. In the present work, as the second direction, we proposed an efficient routing strategy with load-balancing using chaotic neurodynamics for the complex networks.

In our computer network model, a node is taken to be both hosts and routers. In other words, the nodes produce packets and decide routes of the packets. A link serves as a pathway through which packets are transmitted. When a packet is generated at a node, it is transmitted from a node to another through the links. Then, the packet is stored at the tail of the buffer of the transmitted node. All the packets are transmitted according to First-In-First-Out basis. In addition, if the buffer of a node is full, a packet transmitted to the node will be removed. Also, the packet movement is limited by the upper number of hops with which a packet has been routed. Thus, if this limit is exceeded, the packet is also removed. In such a case, the packet is retransmitted from its source until it will be delivered to its destination.

A good routing strategy is to transmit the packets to the destinations as quickly as possible. The Dijkstra algorithm is one of the basic routing strategies which transmits a packet to its destination along with the shortest path to destination of the packet. If all the nodes in the computer network have high performance, that is, the nodes have large buffer sizes and transmit many packets at the same time, this routing strategy works well. However, the buffer size and throughput of each node is different in the real computer network. It means that if one uses the Dijkstra algorithm, the packet congestion occurs in the real computer network. Thus, it is a very important problem to propose a sophisticated routing strategy for avoiding the packet congestion.

To alleviate the packet congestion, one of the possible strategies is not to transmit the packets to an adjacent node to which the packets just have been transmitted for a while. From this view point, we have already proposed a routing strategy with chaotic neurodynamics[1]. In this routing strategy, a refractory effect, which is an important characteristic in nerve membrane[10] and produces the chaotic neurodynamics, memorizes a past routing history. Using the refractory effect or the past routing history, the proposed strategy shows high performance for the small-world type randomized networks and the scale-free networks[1].

In the routing strategy[1], each node determines a transmitted node of the packet using the shortest path information and the chaotic selection. Although the routing strategy shows high performance, if the node obtains additional information, we expect that the performance of it becomes higher. One of the efficient additional information is an

waiting time at an adjacent node or the number of queueing packets at the adjacent node. Thus, in this paper, we improve the previous routing strategy[1]. We confirm that the improved routing strategy has high performance for the complex networks, such as the small-world networks[8] and the scale-free networks[3].

2. A routing method with chaotic neurodynamics

In this section, we explained how to construct the proposed routing strategy. First of all, let us start with constructing a model computer network. The model computer network has N nodes, and the ith node has N_i adjacent nodes ($i = 1, \ldots, N$). Then, we assign a neural network to each node. That is, the ith node has its own neural network which consists of N_i neurons. N_i neurons correspond to N_i adjacent nodes. The firing of the ijth neuron ($j = 1, 2, \ldots, N_i$) encodes the transmission of a packet from the ith node to the jth adjacent node.

In the proposed routing strategy, each node has its own neural network which operates to minimize a distance of the transmitting packet from the ith node to its destination, and an waiting time at the jth adjacent node. To realize this routing strategy, the internal state of the ijth neuron in the neural network is defined as:

$$\xi_{ij}(t+1) = \beta\Big\{H\Big(1 - \frac{d_{ij} + d_{jg(p_i(t))}}{d_c}\Big) + (1-H)\Big(1 - \frac{q_j(t)}{b}\Big)\Big\}, \quad (1)$$

where d_{ij} is a static distance from the ith node to its jth adjacent node; $p_i(t)$ is a transmitted packet of the ith node at the t-th iteration; $g(p_i(t))$ is a destination of $p_i(t)$; $d_{jg(p_i(t))}$ is a dynamic distance from the jth adjacent node to $g(p_i(t))$, that is, $d_{jg(p_i(t))}$ depends on $g(p_i(t))$; d_c is the size of the computer network; β is a normalization parameter; $q_j(t)$ is the number of queueing packets at the jth adjacent node at the t-th iteration; b is the buffer size of each node; H decides priority of the first term and the second term.

If the jth adjacent node is the closest to $g(p_i(t))$, and it has the small number of the queueing packets, $\xi_{ij}(t+1)$ takes the largest value. In the routing strategy[1], the first term of Eq.(1) is only used whether the jth adjacent node is an optimum one or not. The second term of Eq.(1) expresses an waiting time at the jth adjacent node until $p_i(t)$ will be transmitted from the jth adjacent node to the next transmitted node. By adding the waiting time, each node selects the optimum adjacent node more efficiently and flexibly.

Next, we assigned the refractory effect[10] to each neuron. The refractory effect is one of the essential characteristics of a real neuron: a neuron which has just fired hardly fires for a while. In our strategy, we use the refractory effect as a memory effect. Namely, each node can memorize a past routing history using the refractory effect, then, an adjacent node to which many packets have been transmitted is not selected as a transmitted node of the packets for a while. The refractory effect is described as follows:

$$\zeta_{ij}(t+1) = -\alpha \sum_{d=0}^{t} k_r^d x_{ij}(t-d) + \theta, \quad (2)$$

where α is a control parameter of the refractoriness; k_r is a decay parameter of the refractoriness; $x_{ij}(t)$ is the output of the ijth neuron at the t-th iteration that will be defined in Eq.(4); θ is a threshold.

Finally, a mutual connection is assigned to each neuron. The mutual connection controls firing rates of the neurons, because too frequent firing often leads to a fatal situation of the packet routing. The mutual connection is defined as follows:

$$\eta_{ij}(t+1) = W - W \sum_{j=1}^{N_i} x_{ij}(t), \quad (3)$$

where $W > 0$ is a parameter and N_i is the number of adjacent nodes at the ith node.

Then, the output of the ijth neuron is defined as follows:

$$x_{ij}(t+1) = f\{\xi_{ij}(t+1) + \zeta_{ij}(t+1) + \eta_{ij}(t+1)\}, \quad (4)$$

where $f(y) = 1/(1 + e^{-y/\epsilon})$. In this algorithm, if $x_{ij}(t+1) > 1/2$, the ijth neuron fires; the packet at the ith node is transmitted to the jth adjacent node. If the outputs of multiple neurons exceed $1/2$, we defined that the neuron whose output is the largest only fires.

3. Computer simulation

To evaluate the performance of the proposed routing strategy, we compared the proposed routing strategy with three kinds of routing strategies. The first one is the Dijkstra algorithm. The second one is a conventional routing strategy with chaotic neurodynamics (the CNN strategy)[1]. The third one is a routing strategy with a decent down-hill dynamics (which used only in Eq.(1) (the DD strategy)).

The computer simulations are conducted as follows: first, we produced random values from one to five, and assigned them as the throughputs at all nodes. In addition, each node calculates the shortest path from the node to the other nodes. Namely, each node always has a static routing table which contains an information list of the shortest distances between any two nodes. Each packet has a destination and the destination is randomly assigned using uniformly distributed random numbers. Then, at every node, an optimal adjacent node is selected and the packets are simultaneously transmitted to their destinations. The buffer size of every node is set to $1,000$, and the upper number of hops of the packet movement is set to 128. A packet is removed if the packet exceeds the upper number of hops. In such a case, the packet is retransmitted from a source to its destination until it will be delivered to the destination. We conducted 10 simulations to average the results.

We repeated the link selection and packet transmission for $10,000$ iterations. We fixed the total number of packets

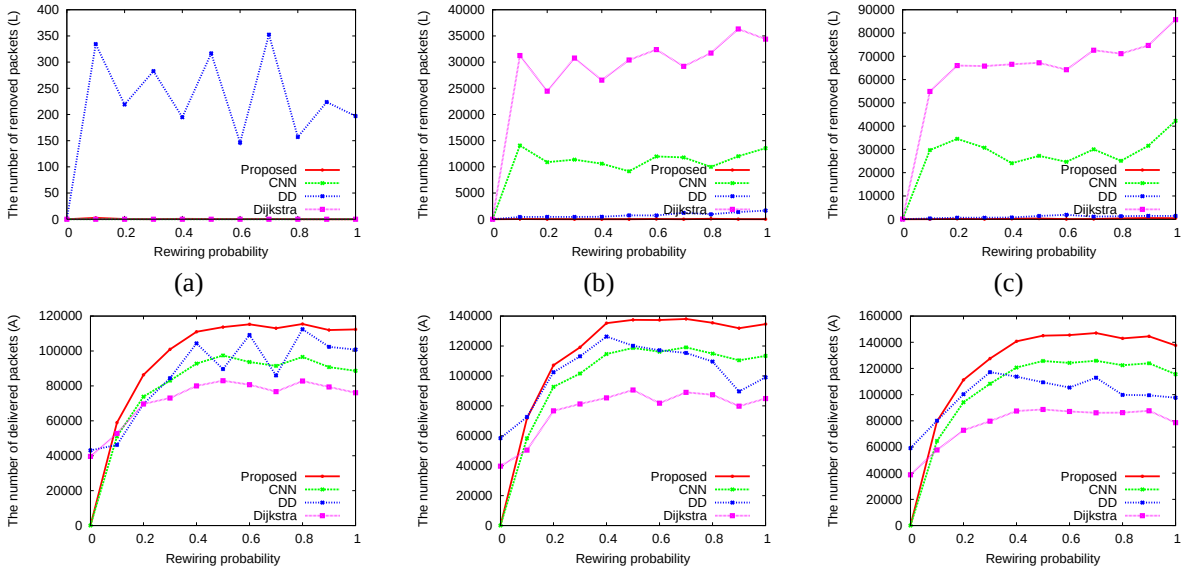

Figure 1: Rewiring probabilities (R_p) and the number of the packets lost before arriving at their destinations(L) for (a) $\overline{S} = 10$, (b) 50, and (c) 100, and the number of delivered packets to their destinations (A) for (c) $\overline{S} = 10$, (d) 50, and (f) 100 for the small-world networks.

in the computer network. Thus, when the packet arrived at its destination, we generated a new packet. Then, a source and its destination of the new packet are randomly decided again using uniformly distributed random numbers. We set the parameters in Eqs.(1)–(4) as follows: $\beta = 1.2$, $H = 0.8$, $\alpha = 0.045$, $k_r = 0.98$, $\theta = 0.5$, $W = 0.05$, and $\epsilon = 0.05$. We also set d_c as a diameter of the computer network, which is defined as the longest distance between two nodes.

As the topologies of the computer networks, 100 nodes of the small-world networks[8] and the scale-free networks[3] are used in these simulations. To evaluate the performance of the proposed strategy and the three conventional routing strategies, we measured an average number of stored packets at each node, $\overline{S}$, the number of packets lost before arriving at their destinations, L, and the number of delivered packets to their destinations, A.

First, we evaluated the proposed strategy, the CNN strategy[1], the DD strategy, and the Dijkstra algorithm for the small-world networks. The small-world network is produced by the same manner as proposed by Watts and Strogatz[8]. First, N nodes are put on a closed one-dimensional ring and each node is connected its K-th nearest neighbors. Then, each link is randomly rewired with probability R_p. We set N and K to 100 and 4, respectively.

Results for the small-world networks are shown in Fig.1. In Figs.1(a), (b) and (c), although the proposed strategy and the DD strategy have almost no removed packets (L) when the average number of stored packets ($\overline{S}$) increases, values of L of the CNN strategy and the Dijkstra algorithm become large. In addition, in Figs.1(d), (e), and (f), the number of delivered packets to their destinations (A) of the proposed strategy becomes larger than the other routing methods when the rewiring probabilities (R_p) increase. In Figs.1(d), (e), and (f), although the number of delivered packets (A) of the CNN strategy is the almost same as that of the DD strategy, the number of removed packets (L) of the CNN strategy is larger than that of the DD strategy. Thus, from the results of Fig.1, the performance of the CNN strategy is worse than the DD strategy.

Next, we evaluated the proposed strategy, the CNN strategy, the DD strategy, and the Dijkstra algorithm for the scale-free networks. The scale-free networks are generated in the same way as Barábasi and Albert[3]. This network is constructed by the following procedure: first, we made a complete graph which has four nodes. Then, we put a new node with three links at every time step. Next, we connected three links of the newly added node to the nodes already existing in the computer network with the probability $\Pi(k_i) = \dfrac{k_i}{\sum_{j=1}^{n} k_j}$, where k_i is the degree of the ith node ($i = 1, \ldots, n$); n is the number of nodes at a current iteration.

Results for the scale-free networks are shown in Fig.2. In Fig.2(a), the number of removed packets (L) of the CNN strategy and the Dijkstra algorithm rapidly increases when the average number of stored packets at each node ($\overline{S}$) becomes large. On the other hand, the proposed strategy and the DD strategy have no removed packet (L) even if $\overline{S}$ becomes large. In addition, in Fig.2(b), the proposed strategy transmits more packets to their destinations when $\overline{S}$ increases comparing with the other routing strategies. In Fig.2(b), as well as the results of the small-world networks (Fig.1), because the number of removed packets (L) of the CNN strategy is larger than that of the DD strategy, the per-

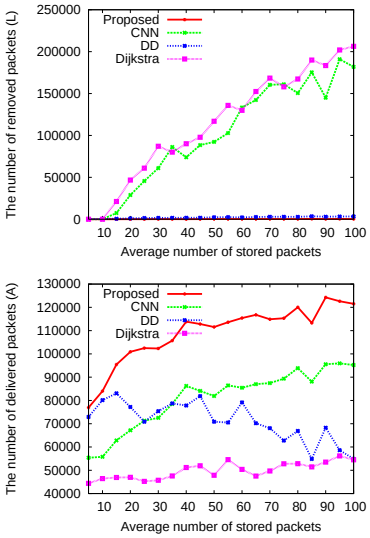

Figure 2: Relationship between an average number of stored packets ($\bar{S}$) and (a) the number of packets lost before arriving at their destinations (L), and (b) the number of packets delivered to their destinations (A) for the scale-free networks.

formance of the CNN strategy is worse than the DD strategy for the scale-free networks.

In Ref.[1], the buffer size of each node is set to the product of $10,000$ and the number of adjacent node at each node. Although the CNN strategy and the Dijkstra algorithm have no number of removed packets (L) using large buffer sizes of the nodes, as seen in the results of the small-world networks (Fig.1) and the scale-free networks (Fig.2), we confirmed that the numbers of the removed packets of the CNN strategy and the Dijkstra algorithm become increases if the buffer size of each node reduced. On the other hand, using the information of the waiting time at the adjacent node, the proposed strategy and the DD strategy effectively transmit the packets to their destinations without loss of the packets. Using large buffer sizes of the nodes corresponds to high spec routers in the real computer network. Thus, from these results, the proposed strategy and the DD strategy show high performance, even if the routers in the computer networks have low performance.

The proposed strategy transmits more packets than the DD strategy for the small-world networks and the scale-free networks. A significant difference between the proposed strategy and the DD strategy exists in the basic dynamics whether the routing strategy has the chaotic neurodynamics or not. Using the chaotic neurodynamics produced by Eq.(2), the proposed strategy decentralized the packets more effectively than the DD strategy for the small-world networks and the scale-free networks, and this efficient decentralization led to high performance of the proposed strategy.

4. Conclusion

In this paper, to improve the performance of the routing strategy with chaotic neurodynamics[1], we introduced the information of the waiting time to transmit the packet at adjacent nodes to each node. Using additional information, the performance is much improved, and the proposed routing strategy has high performance for the small-world networks and the scale-free networks comparing with the previous strategy[1].

Furthermore, using the chaotic neurodynamics, the performance of the proposed routing strategy becomes outstanding when we compare its performance with that of the decent-down hill routing strategy for the small-world networks and the scale-free networks. However, we do not clarify why the chaotic neurodynamics effectively decentralizes the packet congestion in this paper. Thus, in the future works, we consider analysis of the chaotic neurodynamics for effectively decentralization of the packets. One of the solvable methods is to use the method of surrogate data which is an important analysis technique in the field of nonlinear time-series analysis[11] to evaluate the significance of the chaotic neurodynamics.

The research of TI is partially supported by Grant-in-Aids for Scientific Research (B) (No.16300072) and (C) (No.17500136) from JSPS, and a research grant from The Mazda Foundation.

References

[1] T.Kimura and T.Ikeguchi, “A new algorithm for packet routing problems using chaotic neurodynamics and its surrogate analysis,” *Neural computing and applications*, 10.1007/s00521-007-0099-5, 2007.

[2] A.Arenas, A.D´ıaz-Guilera, and R.Guimera, “Communication in networks with hierarchical branching,” *PRL*, vol.86, 3196, 2001.

[3] A.-L.Bar´abasi and R.Albert, “Emergence of scaling in random networks,” *Science*, vol.286, pp.509–512, 1999.

[4] W.Wang, B.Wang, C.Yin, Y.Xie, and T.Zhou, “Traffic dynamics based on local routing protocol on a scale-free network,” *PRE*, vol.73, 026111, 2006.

[5] H.Zhang, Z.Liu, M.Tang, and P.Hui, “An adaptive routing strategy for packet delivery in complex networks,” *PLA*, vol.364, pp.177–182, 2007.

[6] T.Horiguchi and S.Ishioka, “Routing control of packet flow using a neural network,” *Physica A*, vol.297, pp.521–531, 2001.

[7] T.Horiguchi, K.Hayashi and A.Tretiakov, “Reinforcement learning for congestion-avoidance in packet flow,” *Physica A*, vol.349, pp.329–348, 2005.

[8] D.J.Watts and S.H.Strogatz “Collective dynamics of small-world networks,” *Nature*, vol.393, pp.440–442, 1998.

[9] T.Kimura, H.Nakajima and T.Ikeguchi, “A packet routing method for complex networks by a stochastic neural network,” *Physica A*, vol.376, pp.658–672, 2007.

[10] K.Aihara, T.Tanabe, and M.Toyoda, “Chaotic neural network,” *PLA*, vol.144, pp.333–340, 1990.

[11] H.Kantz and T.Schreiber, “Nonlinear time series analysis,” Cambridge University Press, 2003.