國 立 交 通 大 學

應用數學系

碩 士 論 文

廣義的 Shuffle-exchange 網路在三種不同的訊息傳送演算法之效益分析

The Performance Analysis of Three Routing Algorithms of General Shuffle-exchange Networks

研 究 生：林威雄

指導教授：陳秋媛　教授

中 華 民 國 九 十 七 年 一 月

Contents

List of Figures

The Performance Analysis of Three Routing Algorithms of General Shuffle-Exchange Networks

Student: Bruce W. Lin Advisor: Chiuyuan Chen

Department of Applied Mathematics

National Chiao Tung University

Abstract

The shuffle-exchange network is a popular architecture for multistage interconnection networks. The number of nodes in a shuffle-exchange network is usually a power of k if each switching element in the network is of size $k \times k$. In [10], Padmanbhan relaxed the restriction that the number of nodes must be a power of k and proposed the general shuffle-exchange network (GSEN). Padmanbhan also proposed an elegant tag-based routing algorithm for the GSEN. Later, in [2], Chen, Liu, and Qui enhanced the GSEN with bidirectional links and they proposed a tag-based routing algorithm for the backward network of the GSEN. Recently, Chen and Lou [3] also proposed an tag-based routing algorithm for the backward network of the GSEN. A multistage interconnection network enables processors to send their messages concurrently. When two routing requests occur simultaneously in the GSEN, a conflict may occur. The purpose of this thesis is to analyze the performance of the above three tag-based routing algorithms when there are *two routing requests.*

Keywords: multistage interconnection network, parallel and distributed computing, shuffle-exchange network, routing algorithm, conflict.

1 Introduction

The shuffle-exchange network has been proposed as a popular architecture for multistage interconnection networks; see [4, 5, 8, 10, 12]. The number of nodes in a shuffle-exchange network is usually a power of k if each switching element is of size $k \times k$. Since it is desirable to build a multistage interconnection network out of 2×2 switching elements instead of larger switching elements, throughout this thesis, we will assume that all the switching elements are identical and are of size 2×2.

It is well known that a 2×2 switching element has only two possible states: *straight* and *cross*, as shown in Figure 1. As can be seen from Figure 1, a 2×2 switching element has two upper and two lower sub ports. We will use *sub port* 0 (*sub port* 1) denote an upper (a lower) sub port.

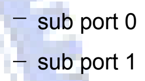

Figure 1: The states of a 2×2 switching element and the sub ports.

In a multistage interconnection network, a path from an input to an output can be described by a sequence of labels that label the successive edges on this path. Such a sequence of labels is called a *control tag* [10] or *tag* [2] or *path descriptor* [6]. The control tag may be used as a header for routing a message: each successive node uses the first element of the sequence to route the message, and then discards it. For example, in Figure 2 (a), input 3 can get to output 4 by using the control tag 6 (0110), which means that the routing is via sub port 0 at stage 0, sub port 1 at stage 1, sub port 1 at stage 2, and sub port 0 at stage 3.

Recall that the number of nodes in a shuffle-exchange network is usually a power of 2 if each switching element is of size 2×2. The general shuffle-exchange network (GSEN) was proposed by Padmanbhan in [10] to relax the restriction on the number of nodes in a shuffle-exchange network. More precisely, an $N' \times N'$ *general shuffle-exchange network* is a multistage interconnection network with N' inputs and N' outputs and each stage

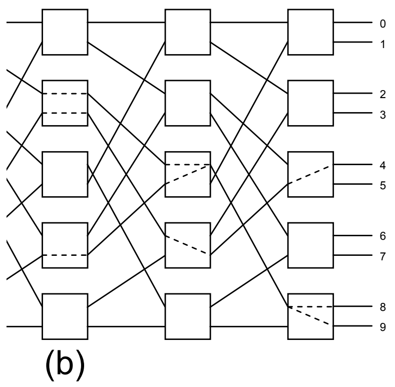

Figure 2: A 10×10 GSEN.

consists of the perfect shuffle operation (defined below) on N' terminals followed by $N'/2$ switching elements. The *perfect shuffle operation* on N' terminals is the permutation π defined by

$$\pi(i) = (2 \cdot i + \left\lfloor \frac{2 \cdot i}{N'} \right\rfloor) \bmod N', \quad 0 \le i \le N' - 1.$$

In other words, the perfect shuffle operation separates the top $N'/2$ terminals from the bottom $N'/2$ terminals and precisely interleaves them, with the bottom terminals still remaining at the bottom. See Figure 2 for an illustration.

Clearly, the number of stages in a GSEN is at least as large as $\log_2 N'$. When it is exactly $\log_2 N'$, the GSEN is identical to the Omega network defined in [7] and the the control tag depends only on the destination. If the number of stages in a GSEN is greater than $\log_2 N'$, the control tag will depend on both the source and the destination. An elegant tag-based routing algorithm for the GSEN has been proposed by Padmanbhan in [10]. In the remaining part of this thesis, we will call this algorithm *P-algorithm* for convenience.

After the work of Padmanbhan [10], Chen et al. [2] enhanced the GSEN with bidirectional links. Their reason for the enhancement is that although unidirectional links are widely used, bidirectional links also have many applications as suggested in [4]. A bidirectional GSEN can be divided into two dependent networks: the *forward network* and the *backward network*. The forward network is from the left-hand side of the GSEN to the right-side of the GSEN; thus a routing request in it is sent from left to right. On the other hand, the backward network is from the right-hand side of the network to the left-hand side of the network; thus a routing request in it is sent from right to left. The control tags used in the forward (backward) network are called the *forward* (*backward*) *control tags*.

Obviously, P-algorithm can be applied on the forward network. As for the backward network, Chen et al. [2] proposed a tag-based routing algorithm for it; this algorithm is based on the idea of inversely using the forward control tag. More precisely, this algorithm first runs P-algorithm to obtain the forward control tag; then, it runs another procedure to convert the forward control tag to the backward control tag. In the remaining part of this thesis, we will call the algorithm of Chen et al. *CLQ-algorithm* for convenience. Recently, Chen and Lou [3] also proposed a tag-based routing algorithm for the backward network. Unlike CLQ-algorithm, Chen and Lou's algorithm does not run P-algorithm first and is not based on the idea of inversely using the forward control tag. In the remaining part of this thesis, we will call Chen and Lou's algorithm *CL-algorithm* for convenience.

A multistage interconnection network enables processors to send their messages concurrently. However, routing must be handled carefully so that there is no conflict when messages are sending concurrently. There are two types of conflict-free routings in a multistage interconnection network: one is *routing with link-disjoint paths* and the other is *routing with node-disjoint paths*. The former is used in an electronic network and the latter, an optical network. Routing with link-disjoint paths means that no two different messages have their paths share the same link in the network, while routing with node-disjoint paths means that no two different messages have their paths share the same switching element in the network.

The three known routing algorithms for the GSEN are P-algorithm, CLQ-algorithm, and CL-algorithm. P-algorithm can be used to send a message in a GSEN or in the forward network of a bidirectional GSEN. Both CLQ-algorithm and CL-algorithm can be used to send a message in the backward network of a bidirectional GSEN. When two routing requests occur simultaneously, a conflict may occur. Up to now, there is no analysis for the conflicts of routing requests in the GSEN. For some routing requests, P-algorithm and CLQ-algorithm can provide two control tags, i.e., two paths that can fulfill the routing request. On the other hand, for each routing request, CL-algorithm provides only one control tag, i.e., only one path that can fulfill the routing request. It is also not known which control tag should be used (when two tags are available) to reduce the conflict. The purpose of this thesis is to analyze the performance of the above three routing algorithms of the GSEN. We will focus on the case that there are *two* routing requests.

This thesis is organized as follows: Section 2 gives some preliminaries and the idea of our performance analysis. Sections 3, 4, and 5 give the analysis of P-algorithm, CLQ-algorithm, and CL-algorithm, respectively. Concluding remarks are given in the final section.

2 Preliminaries and the idea of our analysis

We first introduce some terminologies that will be used throughout this thesis. For convenience, GSEN is also used to denote a bidirectional GSEN. N' is used to denote the number of inputs and outputs of the given GSEN. Also, r and $n+1$ are used to denote the number of switching elements in a stage and the number of stages in a GSEN, respectively. A GSEN is a multistage interconnection network with switches *aligned* in $n+1$ stages, labelled $0, 1, \ldots, n$. Each stage consists of r switching elements, labelled with $0, 1, \ldots, r-1$. Since every switching element is of size 2×2, there are a total of

$$N' = 2 \times r$$

ports on each side of a stage, labelled $0, 1, \cdots, N'-1$. The parameters N', r, and n satisfy the following equation:

$$\lceil \log_2(2 \cdot r) \rceil = \lceil \log_2 N' \rceil = n + 1.$$

The following conventions are used in this thesis. Stage 0 is the leftmost stage even if the network is the backward network of a GSEN. The switching elements in a stage are considered cyclic; that is, the switching element labelled 0 is considered to be the successive switch element of the switching element labelled $r-1$. Also, nodes i and i' (j and j') are assumed to be on the left-hand (right-hand) side of the network. An (i,j)-*request* denotes a request for sending a message from i to j (from j to i if the network is a backward network). An (i,j)-*path* denotes a path between i and j. The terms (i',j')-*request* and (i',j')-*path* are defined similarly.

Note that an (i,j)-request can be fulfilled by an (i,j)-path. Moreover, an (i,j)-path can be characterized by its *port sequence*, which is the sequence of ports

$$(R_{-1}, R_0, R_1, \cdots, R_n)$$

passed by this path such that R_{-1} is defined to be

$$R_{-1} = i$$

and R_ℓ is the port to the right of the switching element at stage ℓ on this path. Clearly,

$$R_n = j.$$

Take Figure 2 (a) for an example. A $(3,4)$-request can be fulfilled by using the $(3,4)$-path shown in this figure and this $(3,4)$-path has the port sequence $(3,6,3,7,4)$. The purpose of this thesis is to analyze the performance of three existing algorithms of GSENs: P-algorithm, CLQ-algorithm, and CL-algorithm. And we will focus on the case that two routing requests occur simultaneously. In an electronic GSEN, two routing requests can be sent simultaneously if their routing paths are *link-disjoint*, meaning that no two links of these two paths are identical. On the other hand, in an optical GSEN, two routing requests can be sent simultaneously if their routing paths are *node-disjoint*, meaning that no two switching elements of these two paths are identical; this is to ensure that only one signal passes through a switching element at a time and thus to avoid the *crosstalk problem* (see also [16]).

Two routing paths are said to have a *link-conflict* (*node-conflict*) if they are not link-disjoint (node-disjoint). The following two lemmas are obvious and their proofs are omitted.

Lemma 1. *If two routing paths have a link-conflict, then they have a node-conflict.*

Lemma 2. *There are only three possible cases for two routing paths:*

(i) they have no node-conflict (hence no link-conflict);

(ii) they have a node-conflict and have no link-conflict;

(iii) they have a link-conflict (hence a node-conflict).

Take Figure 2 (b) for an example. Suppose we have three routing requests: $(3, 4)$-request, $(0, 8)$-request, and $(9, 9)$-request. Also suppose that these three requests are routed along path $\mathcal{P}$ with port sequence $(3, 6, 3, 7, 4)$, path $\mathcal{Q}$ with port sequence $(0, 1, 2, 4, 8)$, and path $\mathcal{U}$ with port sequence $(9, 8, 7, 4, 9)$. Then $\mathcal{P}$ and $\mathcal{U}$ have no node-conflict and no link-conflict; $\mathcal{P}$ and $\mathcal{Q}$ have a node-conflict and have no link-conflict; $\mathcal{U}$ and $\mathcal{Q}$ have a link-conflict and a node-conflict.

We now describe the idea used in our analysis. Suppose the two requests that occur simultaneously are (i, j)-request and (i', j')-request. Also suppose that (i, j)-request is routed along the (i, j)-path $\mathcal{P}$ with port sequence

$$(\mathcal{P}_{-1}, \mathcal{P}_0, \mathcal{P}_1, \cdots, \mathcal{P}_n) \tag{1}$$

and (i', j')-request are routed along the (i', j')-path $\mathcal{Q}$ with port sequence

$$(\mathcal{Q}_{-1}, \mathcal{Q}_0, \mathcal{Q}_1, \cdots, \mathcal{Q}_n). \tag{2}$$

The following two lemmas describe how we detect a link-conflict or a node-conflict.

Lemma 3. *$\mathcal{P}$ and $\mathcal{Q}$ have a link-conflict if and only if*

$$\mathcal{P}_k = \mathcal{Q}_k \text{ for some } -1 \leq k \leq n.$$

Proof. For each $-1 \leq k \leq n$, port $\mathcal{P}_k$ determines a link on $\mathcal{P}$. Similarly, for each $-1 \leq k \leq n$, port Q_k determines a link on $\mathcal{Q}$. Thus we have this lemma. ∎

Lemma 4. *$\mathcal{P}$ and $\mathcal{Q}$ have a have a node-conflict if and only if*

$$\lfloor \frac{\mathcal{P}_k}{2} \rfloor = \lfloor \frac{\mathcal{Q}_k}{2} \rfloor \text{ for some } 0 \leq k \leq n.$$

Proof. For each $0 \le k \le n$, $\lfloor \frac{\mathcal{P}_k}{2} \rfloor$ determines a switching element on $\mathcal{P}$. Similarly, for each $0 \le k \le n$, $\lfloor \frac{\mathcal{Q}_k}{2} \rfloor$ determines a switching element on $\mathcal{Q}$. Thus we have this lemma. ∎

In the following sections, we will further assume that

$$i \neq i' \text{ and } j \neq j'.$$

This is because if $i = i'$ or $j = j'$ occur, then a link-conflict and also a node-conflict will occur and it is definitely impossible to fulfill these two routing requests simultaneously.

3 The analysis of P-algorithm

In this section, we will analyze the performance of P-algorithm. This algorithm was stated in a theorem in [10].

Theorem 5. [10] *Any i, $0 \le i < N'$, in a GSEN can set up a path to a j, $0 \le j < N'$, by using the control tag*

$$T_1 = (j + 2Mi) \bmod N'.$$

In addition if $T_1 + N' < 2N$, then a second control tag exists and is given by

$$T_2 = T_1 + N'.$$

Using P-algorithm, each of the two requests (i, j)-request and (i', j')-request can be fulfilled by using a T_1 or a T_2 control tag. Note that the second control tag T_2 may not exist. In this occurs, we will set $T_2 = T_1$. Thus there are four possible cases:

T_1T_1**-case:** Both requests are fulfilled by using their T_1 tags.

T_1T_2**-case:** (i, j)-request is fulfilled by using its T_1 tag and (i', j')-request, its T_2 tag.

T_2T_1**-case:** (i, j)-request is fulfilled by using its T_2 tag and (i', j')-request, its T_1 tag.

T_2T_2**-case:** Both requests are fulfilled by using their T_2 tags.

We run computer programs to obtain the number of node-conflicts and the number of link-conflicts. For convenience, let

$$x \in \{T_1T_1, T_1T_2, T_2T_1, T_2T_2\}. \tag{3}$$

Let LCF(i, j, x) denote the number of (i', j')-requests that have a link-conflict with the (i, j)-request when the x-case occurs. Let NCF(i, j, x) denote the number of (i', j')-requests that have a node-conflict with the (i, j)-request when the x-case occurs. For example, NCF(i, j, T_1T_2) denotes the number of (i', j')-requests that have a node-conflict with the (i, j)-request when the T_1T_2-case occurs. To describe the results of our analysis, we also define

$$\text{TotalLCF}(i, *, x) = \sum_{j=0}^{N'-1} \text{LCF}(i, j, x),$$

$$\text{TotalNCF}(i, *, x) = \sum_{j=0}^{N'-1} \text{NCF}(i, j, x),$$

$$\text{TotalLCF}(*, j, x) = \sum_{i=0}^{N'-1} \text{LCF}(i, j, x),$$

$$\text{TotalNCF}(*, j, x) = \sum_{i=0}^{N'-1} \text{NCF}(i, j, x).$$

For convenience, if a (i, j)-path is obtained from control tag T_1, then we say it is a T_1-(i, j)-path and if it is obtained from control tag T_2, then we say it is a T_2-(i, j)-path. We have two lemmas.

Lemma 6. *If there is a T_ℓ-(i, j)-path with port sequence $(i, R_0, R_1, R_2, \cdots, R_{n-1}, j)$, then there is a $T_{3-\ell}$-$(N' - 1 - i, N' - 1 - j)$-path with port sequence $(N' - 1 - i, N' - 1 - R_0,$ $N' - 1 - R_1, N' - 1 - R_2, \cdots, N' - 1 - R_{n-1}, N' - 1 - j)$ for $\ell = 1, 2$.*

Proof. This lemma follows from the fact that if we fold a GSEN so that its upper boundary coincides with its lower boundary, then a T_ℓ-(i, j)-path with port sequence $(i, R_0, R_1, R_2, \cdots, R_{n-1}, j)$ will coincide with a $T_{3-\ell}$-$(N' - 1 - i, N' - 1 - j)$-path with port sequence $(N' - 1 - i, N' - 1 - R_0, N' - 1 - R_1, N' - 1 - R_2, \cdots, N' - 1 - R_{n-1}, N' - 1 - j)$.

∎

Lemma 7. *If there is link-conflict between the T_ℓ-(i, j)-path and the T_m-(i', j')-path, then there is link-conflict between the $T_{3-\ell}$-$(N'-1-i, N'-1-j)$-path and the T_{3-m}-$(N'-1-i', N'-1-j')$-path for $\ell = 1, 2$ and $m = 1, 2$.*

Proof. Let the port sequences of the T_ℓ-(i, j)-path $\mathcal{P}$ and the T_m-(i', j')-path $\mathcal{Q}$ be $(i, \mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_{n-1}, j)$ and $(i', \mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2, \cdots, \mathcal{Q}_{n-1}, j')$, respectively. By Lemma 3, there exists $-1 \le k \le n$ such that $\mathcal{P}_k = \mathcal{Q}_k$, which implies $N' - 1 - \mathcal{P}_k = N' - 1 - \mathcal{Q}_k$. This lemma now follows from Lemmas 3 and 6. ∎

The above two lemmas lead to the following two corollaries.

Corollary 8.

$$\begin{array}{lrcl}
\text{(A)} & \mathit{LCF}(i, j, T_1T_1) & = & \mathit{LCF}(N'-1-i, N'-1-j, T_2T_2) \\
\text{(B)} & \mathit{NCF}(i, j, T_1T_1) & = & \mathit{NCF}(N'-1-i, N'-1-j, T_2T_2) \\
\text{(C)} & \mathit{LCF}(i, j, T_1T_2) & = & \mathit{LCF}(N'-1-i, N'-1-j, T_2T_1) \\
\text{(D)} & \mathit{NCF}(i, j, T_1T_2) & = & \mathit{NCF}(N'-1-i, N'-1-j, T_2T_1)
\end{array}$$

Proof. The first statement of this corollary follows from Lemma 7. The other statements of this corollary can be proven in a similar way and we omit their proofs. ∎

Corollary 9.

$$\begin{array}{lrcl}
\text{(a)} & \mathit{TotalLCF}(i, *, T_1T_1) & = & \mathit{TotalLCF}(N'-1-i, *, T_2T_2) \\
\text{(b)} & \mathit{TotalLCF}(*, j, T_1T_1) & = & \mathit{TotalLCF}(*, N'-1-j, T_2T_2) \\
\text{(c)} & \mathit{TotalNCF}(i, *, T_1T_1) & = & \mathit{TotalNCF}(N'-1-i, *, T_2T_2) \\
\text{(d)} & \mathit{TotalNCF}(*, j, T_1T_1) & = & \mathit{TotalNCF}(*, N'-1-j, T_2T_2) \\
\text{(e)} & \mathit{TotalLCF}(i, *, T_1T_2) & = & \mathit{TotalLCF}(N'-1-i, *, T_2T_1) \\
\text{(f)} & \mathit{TotalLCF}(*, j, T_1T_2) & = & \mathit{TotalLCF}(*, N'-1-j, T_2T_1) \\
\text{(g)} & \mathit{TotalNCF}(i, *, T_1T_2) & = & \mathit{TotalNCF}(N'-1-i, *, T_2T_1) \\
\text{(h)} & \mathit{TotalNCF}(*, j, T_1T_2) & = & \mathit{TotalNCF}(*, N'-1-j, T_2T_1)
\end{array}$$

Proof.

(a) holds since:

$$\begin{aligned}
\mathrm{TotalLCF}(i, *, T_1T_1) &= \sum_{j=0}^{N'-1} \mathrm{LCF}(i, j, T_1T_1) \quad \text{(by definition)} \\
&= \sum_{j=0}^{N'-1} \mathrm{LCF}(N'-1-i, N'-1-j, T_2T_2) \quad \text{(by (A) of Corollary 8)} \\
&= \mathrm{TotalLCF}(N'-1-i, *, T_2T_2) \quad \text{(by definition).}
\end{aligned}$$

(b) holds since:

$$\begin{aligned}
\text{TotalLCF}(*, j, T_1T_1) &= \sum_{i=0}^{N'-1} \text{LCF}(i, j, T_1T_1) \quad \text{(by definition)} \\
&= \sum_{i=0}^{N'-1} \text{LCF}(N'-1-i, N'-1-j, T_2T_2) \quad \text{(by (A) of Corollary 8)} \\
&= \text{TotalLCF}(*, N'-1-i, T_2T_2) \quad \text{(by definition)}.
\end{aligned}$$

(c) and (d) can be proven in a similar way except (B), instead of (A), of Corollary 8 is used. (e) and (f) can be proven in a similar way except (C), instead of (A), of Corollary 8 is used. (g) and (h) can also be proven in a similar way except (D), instead of (A), of Corollary 8 is used. ∎

To obtain LCF(i, j, x) or NCF(i, j, x), we need to test $(N'-1) \times (N'-1)$ pairs of (i', j')-requests. We have run computer programs for each $N' = 4, 6, \cdots, 46$ and have obtained the following values:

LCF(i, j, x) for all i, j, x; TotalLCF$(i, *, x)$ for all i, x; TotalLCF$(*, j, x)$ for all j, x;

NCF(i, j, x) for all i, j, x; TotalNCF$(i, *, x)$ for all i, x; TotalNCF$(*, j, x)$ for all j, x.

From our computer output, we have the following observations.

Observation 1. For each $x \in \{T_1T_1, T_2T_2, T_1T_2, T_2T_1\}$,

$$\text{LCF}(i, j, x) = \text{LCF}(0, j + 2Mi, x),$$

$$\text{NCF}(i, j, x) = \text{NCF}(0, j + 2Mi, x).$$

Observation 2. For each $x \in \{T_1T_1, T_2T_2, T_1T_2, T_2T_1\}$,

$$\text{TotalLCF}(i_1, *, x) = \text{TotalLCF}(i_2, *, x) = \text{TotalLCF}(*, j_1, x) = \text{TotalLCF}(*, j_2, x),$$

$$\text{TotalNCF}(i_1, *, x) = \text{TotalNCF}(i_2, *, x) = \text{TotalNCF}(*, j_1, x) = \text{TotalNCF}(*, j_2, x).$$

Observation 3. TotalLCF$(i, *, T_1T_1)$ = TotalLCF$(i, *, T_1T_2)$ whenever $N' = 2^{n+1}$.

Observation 4. TotalNCF$(i, *, T_1T_1)$ = TotalNCF$(i, *, T_1T_2)$ whenever $N' = 2^{n+1}$.

Observation 5. TotalLCF$(i, *, T_1T_1) >$ TotalLCF$(i, *, T_1T_2)$ whenever $N' \neq 2^{n+1}$.

Observation 6. $\mathrm{TotalNCF}(i,*,T_1T_1) > \mathrm{TotalNCF}(i,*,T_1T_2)$ whenever $N' \neq 2^{n+1}$ and $4|N'$.

Observation 7. $\mathrm{TotalNCF}(i,*,T_1T_1) < \mathrm{TotalNCF}(i,*,T_1T_2)$ whenever $N' \neq 2^{n+1}$ and $4 \not| N'$.

From the above, we also have the following equalities.

$$\begin{aligned}
\mathrm{TotalLCF}(i,*,T_1T_1) &= \mathrm{TotalLCF}(0,*,T_1T_1) && \text{(by Observation 2)}\\
\mathrm{TotalLCF}(*,j,T_1T_1) &= \mathrm{TotalLCF}(0,*,T_1T_1) && \text{(by Observation 2)}\\
\mathrm{TotalLCF}(i,*,T_2T_2) &= \mathrm{TotalLCF}(0,*,T_1T_1) && \text{(by Observation 2 and Corollary 9)}\\
\mathrm{TotalLCF}(*,j,T_2T_2) &= \mathrm{TotalLCF}(0,*,T_1T_1) && \text{(by Observation 2 and Corollary 9)}\\
\mathrm{TotalNCF}(i,*,T_1T_1) &= \mathrm{TotalNCF}(0,*,T_1T_1) && \text{(by Observation 2)}\\
\mathrm{TotalNCF}(*,j,T_1T_1) &= \mathrm{TotalNCF}(0,*,T_1T_1) && \text{(by Observation 2)}\\
\mathrm{TotalNCF}(i,*,T_2T_2) &= \mathrm{TotalNCF}(0,*,T_1T_1) && \text{(by Observation 2 and Corollary 9)}\\
\mathrm{TotalNCF}(*,j,T_2T_2) &= \mathrm{TotalNCF}(0,*,T_1T_1) && \text{(by Observation 2 and Corollary 9)}\\
\mathrm{TotalLCF}(i,*,T_1T_2) &= \mathrm{TotalLCF}(0,*,T_1T_2) && \text{(by Observation 2)}\\
\mathrm{TotalLCF}(*,j,T_1T_2) &= \mathrm{TotalLCF}(0,*,T_1T_2) && \text{(by Observation 2)}\\
\mathrm{TotalLCF}(i,*,T_2T_1) &= \mathrm{TotalLCF}(0,*,T_1T_2) && \text{(by Observation 2 and Corollary 9)}\\
\mathrm{TotalLCF}(*,j,T_2T_1) &= \mathrm{TotalLCF}(0,*,T_1T_2) && \text{(by Observation 2 and Corollary 9)}\\
\mathrm{TotalNCF}(i,*,T_1T_2) &= \mathrm{TotalNCF}(0,*,T_1T_2) && \text{(by Observation 2)}\\
\mathrm{TotalNCF}(*,j,T_1T_2) &= \mathrm{TotalNCF}(0,*,T_1T_2) && \text{(by Observation 2)}\\
\mathrm{TotalNCF}(i,*,T_2T_1) &= \mathrm{TotalNCF}(0,*,T_1T_2) && \text{(by Observation 2 and Corollary 9)}\\
\mathrm{TotalNCF}(*,j,T_2T_1) &= \mathrm{TotalNCF}(0,*,T_1T_2) && \text{(by Observation 2 and Corollary 9)}
\end{aligned}$$

In the following, we list the computer output when $N' = 18$. The values of each $\mathrm{LCF}(i,j,T_1T_1)$, $\mathrm{LCF}(i,j,T_1T_2)$, $\mathrm{LCF}(i,j,T_2T_1)$, $\mathrm{LCF}(i,j,T_2T_2)$, $\mathrm{NCF}(i,j,T_1T_1)$, $\mathrm{NCF}(i,j,T_1T_2)$, $\mathrm{NCF}(i,j,T_2T_1)$, and $\mathrm{NCF}(i,j,T_2T_2)$ are listed in Tables 1 to 8, respectively.

Table 1: Each (i, j)-entry in the table is $\mathrm{LCF}(i, j, T_1T_1)$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
0	27	27	29	29	31	31	29	29	27	27	29	29	27	27	25	25	25	25
1	31	31	29	29	27	27	29	29	27	27	25	25	25	25	27	27	29	29
2	27	27	29	29	27	27	25	25	25	25	27	27	29	29	31	31	29	29
3	27	27	25	25	25	25	27	27	29	29	31	31	29	29	27	27	29	29
4	25	25	27	27	29	29	31	31	29	29	27	27	29	29	27	27	25	25
5	29	29	31	31	29	29	27	27	29	29	27	27	25	25	25	25	27	27
6	29	29	27	27	29	29	27	27	25	25	25	25	27	27	29	29	31	31
7	29	29	27	27	25	25	25	25	27	27	29	29	31	31	29	29	27	27
8	25	25	25	25	27	27	29	29	31	31	29	29	27	27	29	29	27	27
9	27	27	29	29	31	31	29	29	27	27	29	29	27	27	25	25	25	25
10	31	31	29	29	27	27	29	29	27	27	25	25	25	25	27	27	29	29
11	27	27	29	29	27	27	25	25	25	25	27	27	29	29	31	31	29	29
12	27	27	25	25	25	25	27	27	29	29	31	31	29	29	27	27	29	29
13	25	25	27	27	29	29	31	31	29	29	27	27	29	29	27	27	25	25
14	29	29	31	31	29	29	27	27	29	29	27	27	25	25	25	25	27	27
15	29	29	27	27	29	29	27	27	25	25	25	25	27	27	29	29	31	31
16	29	29	27	27	25	25	25	25	27	27	29	29	31	31	29	29	27	27
17	25	25	25	25	27	27	29	29	31	31	29	29	27	27	29	29	27	27

Table 2: Each (i, j)-entry in the table is $\mathrm{LCF}(i, j, T_1T_2)$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
0	26	26	28	28	30	30	28	28	26	26	28	28	26	26	25	25	25	25
1	30	30	28	28	26	26	28	28	26	26	25	25	25	25	26	26	28	28
2	26	26	28	28	26	26	25	25	25	25	26	26	28	28	30	30	28	28
3	26	26	25	25	25	25	26	26	28	28	30	30	28	28	26	26	28	28
4	25	25	26	26	28	28	30	30	28	28	26	26	28	28	26	26	25	25
5	28	28	30	30	28	28	26	26	28	28	26	26	25	25	25	25	26	26
6	28	28	26	26	28	28	26	26	25	25	25	25	26	26	28	28	30	30
7	28	28	26	26	25	25	25	25	26	26	28	28	30	30	28	28	26	26
8	25	25	25	25	26	26	28	28	30	30	28	28	26	26	28	28	26	26
9	26	26	28	28	30	30	28	28	26	26	28	28	26	26	25	25	25	25
10	30	30	28	28	26	26	28	28	26	26	25	25	25	25	26	26	28	28
11	26	26	28	28	26	26	25	25	25	25	26	26	28	28	30	30	28	28
12	26	26	25	25	25	25	26	26	28	28	30	30	28	28	26	26	28	28
13	25	25	26	26	28	28	30	30	28	28	26	26	28	28	26	26	25	25
14	28	28	30	30	28	28	26	26	28	28	26	26	25	25	25	25	26	26
15	28	28	26	26	28	28	26	26	25	25	25	25	26	26	28	28	30	30
16	28	28	26	26	25	25	25	25	26	26	28	28	30	30	28	28	26	26
17	25	25	25	25	26	26	28	28	30	30	28	28	26	26	28	28	26	26

Table 3: Each (i, j)-entry in the table is $\mathrm{LCF}(i, j, T_2T_1)$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
0	26	26	28	28	26	26	28	28	30	30	28	28	26	26	25	25	25	25
1	26	26	28	28	30	30	28	28	26	26	25	25	25	25	26	26	28	28
2	30	30	28	28	26	26	25	25	25	25	26	26	28	28	26	26	28	28
3	26	26	25	25	25	25	26	26	28	28	26	26	28	28	30	30	28	28
4	25	25	26	26	28	28	26	26	28	28	30	30	28	28	26	26	25	25
5	28	28	26	26	28	28	30	30	28	28	26	26	25	25	25	25	26	26
6	28	28	30	30	28	28	26	26	25	25	25	25	26	26	28	28	26	26
7	28	28	26	26	25	25	25	25	26	26	28	28	26	26	28	28	30	30
8	25	25	25	25	26	26	28	28	26	26	28	28	30	30	28	28	26	26
9	26	26	28	28	26	26	28	28	30	30	28	28	26	26	25	25	25	25
10	26	26	28	28	30	30	28	28	26	26	25	25	25	25	26	26	28	28
11	30	30	28	28	26	26	25	25	25	25	26	26	28	28	26	26	28	28
12	26	26	25	25	25	25	26	26	28	28	26	26	28	28	30	30	28	28
13	25	25	26	26	28	28	26	26	28	28	30	30	28	28	26	26	25	25
14	28	28	26	26	28	28	30	30	28	28	26	26	25	25	25	25	26	26
15	28	28	30	30	28	28	26	26	25	25	25	25	26	26	28	28	26	26
16	28	28	26	26	25	25	25	25	26	26	28	28	26	26	28	28	30	30
17	25	25	25	25	26	26	28	28	26	26	28	28	30	30	28	28	26	26

Table 4: Each (i, j)-entry in the table is $\mathrm{LCF}(i, j, T_2T_2)$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
0	27	27	29	29	27	27	29	29	31	31	29	29	27	27	25	25	25	25
1	27	27	29	29	31	31	29	29	27	27	25	25	25	25	27	27	29	29
2	31	31	29	29	27	27	25	25	25	25	27	27	29	29	27	27	29	29
3	27	27	25	25	25	25	27	27	29	29	27	27	29	29	31	31	29	29
4	25	25	27	27	29	29	27	27	29	29	31	31	29	29	27	27	25	25
5	29	29	27	27	29	29	31	31	29	29	27	27	25	25	25	25	27	27
6	29	29	31	31	29	29	27	27	25	25	25	25	27	27	29	29	27	27
7	29	29	27	27	25	25	25	25	27	27	29	29	27	27	29	29	31	31
8	25	25	25	25	27	27	29	29	27	27	29	29	31	31	29	29	27	27
9	27	27	29	29	27	27	29	29	31	31	29	29	27	27	25	25	25	25
10	27	27	29	29	31	31	29	29	27	27	25	25	25	25	27	27	29	29
11	31	31	29	29	27	27	25	25	25	25	27	27	29	29	27	27	29	29
12	27	27	25	25	25	25	27	27	29	29	27	27	29	29	31	31	29	29
13	25	25	27	27	29	29	27	27	29	29	31	31	29	29	27	27	25	25
14	29	29	27	27	29	29	31	31	29	29	27	27	25	25	25	25	27	27
15	29	29	31	31	29	29	27	27	25	25	25	25	27	27	29	29	27	27
16	29	29	27	27	25	25	25	25	27	27	29	29	27	27	29	29	31	31
17	25	25	25	25	27	27	29	29	27	27	29	29	31	31	29	29	27	27

Table 5: Each (i, j)-entry in the table is $\mathrm{NCF}(i, j, T_1T_1)$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
0	53	53	57	57	57	57	61	61	57	57	61	61	61	61	65	65	81	81
1	57	57	61	61	57	57	61	61	61	61	65	65	81	81	53	53	57	57
2	57	57	61	61	61	61	65	65	81	81	53	53	57	57	57	57	61	61
3	61	61	65	65	81	81	53	53	57	57	57	57	61	61	57	57	61	61
4	81	81	53	53	57	57	57	57	61	61	57	57	61	61	61	61	65	65
5	57	57	57	57	61	61	57	57	61	61	61	61	65	65	81	81	53	53
6	61	61	57	57	61	61	61	61	65	65	81	81	53	53	57	57	57	57
7	61	61	61	61	65	65	81	81	53	53	57	57	57	57	61	61	57	57
8	65	65	81	81	53	53	57	57	57	57	61	61	57	57	61	61	61	61
9	53	53	57	57	57	57	61	61	57	57	61	61	61	61	65	65	81	81
10	57	57	61	61	57	57	61	61	61	61	65	65	81	81	53	53	57	57
11	57	57	61	61	61	61	65	65	81	81	53	53	57	57	57	57	61	61
12	61	61	65	65	81	81	53	53	57	57	57	57	61	61	57	57	61	61
13	81	81	53	53	57	57	57	57	61	61	57	57	61	61	61	61	65	65
14	57	57	57	57	61	61	57	57	61	61	61	61	65	65	81	81	53	53
15	61	61	57	57	61	61	61	61	65	65	81	81	53	53	57	57	57	57
16	61	61	61	61	65	65	81	81	53	53	57	57	57	57	61	61	57	57
17	65	65	81	81	53	53	57	57	57	57	61	61	57	57	61	61	61	61

Table 6: Each (i, j)-entry in the table is $\mathrm{NCF}(i, j, T_1T_2)$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
0	89	89	85	85	85	85	81	81	85	85	81	81	81	81	81	81	65	65
1	85	85	81	81	85	85	81	81	81	81	81	81	65	65	89	89	85	85
2	85	85	81	81	81	81	81	81	65	65	89	89	85	85	85	85	81	81
3	81	81	81	81	65	65	89	89	85	85	85	85	81	81	85	85	81	81
4	65	65	89	89	85	85	85	85	81	81	85	85	81	81	81	81	81	81
5	85	85	85	85	81	81	85	85	81	81	81	81	81	81	65	65	89	89
6	81	81	85	85	81	81	81	81	81	81	65	65	89	89	85	85	85	85
7	81	81	81	81	81	81	65	65	89	89	85	85	85	85	81	81	85	85
8	81	81	65	65	89	89	85	85	85	85	81	81	85	85	81	81	81	81
9	89	89	85	85	85	85	81	81	85	85	81	81	81	81	81	81	65	65
10	85	85	81	81	85	85	81	81	81	81	81	81	65	65	89	89	85	85
11	85	85	81	81	81	81	81	81	65	65	89	89	85	85	85	85	81	81
12	81	81	81	81	65	65	89	89	85	85	85	85	81	81	85	85	81	81
13	65	65	89	89	85	85	85	85	81	81	85	85	81	81	81	81	81	81
14	85	85	85	85	81	81	85	85	81	81	81	81	81	81	65	65	89	89
15	81	81	85	85	81	81	81	81	81	81	65	65	89	89	85	85	85	85
16	81	81	81	81	81	81	65	65	89	89	85	85	85	85	81	81	85	85
17	81	81	65	65	89	89	85	85	85	85	81	81	85	85	81	81	81	81

Table 7: Each (i, j)-entry in the table is $\mathrm{NCF}(i, j, T_2T_1)$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
0	81	81	81	81	85	85	81	81	85	85	85	85	89	89	65	65	81	81
1	85	85	81	81	85	85	85	85	89	89	65	65	81	81	81	81	81	81
2	85	85	85	85	89	89	65	65	81	81	81	81	81	81	85	85	81	81
3	89	89	65	65	81	81	81	81	81	81	85	85	81	81	85	85	85	85
4	81	81	81	81	81	81	85	85	81	81	85	85	85	85	89	89	65	65
5	81	81	85	85	81	81	85	85	85	85	89	89	65	65	81	81	81	81
6	81	81	85	85	85	85	89	89	65	65	81	81	81	81	81	81	85	85
7	85	85	89	89	65	65	81	81	81	81	81	81	85	85	81	81	85	85
8	65	65	81	81	81	81	81	81	85	85	81	81	85	85	85	85	89	89
9	81	81	81	81	85	85	81	81	85	85	85	85	89	89	65	65	81	81
10	85	85	81	81	85	85	85	85	89	89	65	65	81	81	81	81	81	81
11	85	85	85	85	89	89	65	65	81	81	81	81	81	81	85	85	81	81
12	89	89	65	65	81	81	81	81	81	81	85	85	81	81	85	85	85	85
13	81	81	81	81	81	81	85	85	81	81	85	85	85	85	89	89	65	65
14	81	81	85	85	81	81	85	85	85	85	89	89	65	65	81	81	81	81
15	81	81	85	85	85	85	89	89	65	65	81	81	81	81	81	81	85	85
16	85	85	89	89	65	65	81	81	81	81	81	81	85	85	81	81	85	85
17	65	65	81	81	81	81	81	81	85	85	81	81	85	85	85	85	89	89

Table 8: Each (i, j)-entry in the table is $\mathrm{NCF}(i, j, T_2T_2)$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
0	61	61	61	61	57	57	61	61	57	57	57	57	53	53	81	81	65	65
1	57	57	61	61	57	57	57	57	53	53	81	81	65	65	61	61	61	61
2	57	57	57	57	53	53	81	81	65	65	61	61	61	61	57	57	61	61
3	53	53	81	81	65	65	61	61	61	61	57	57	61	61	57	57	57	57
4	65	65	61	61	61	61	57	57	61	61	57	57	57	57	53	53	81	81
5	61	61	57	57	61	61	57	57	57	57	53	53	81	81	65	65	61	61
6	61	61	57	57	57	57	53	53	81	81	65	65	61	61	61	61	57	57
7	57	57	53	53	81	81	65	65	61	61	61	61	57	57	61	61	57	57
8	81	81	65	65	61	61	61	61	57	57	61	61	57	57	57	57	53	53
9	61	61	61	61	57	57	61	61	57	57	57	57	53	53	81	81	65	65
10	57	57	61	61	57	57	57	57	53	53	81	81	65	65	61	61	61	61
11	57	57	57	57	53	53	81	81	65	65	61	61	61	61	57	57	61	61
12	53	53	81	81	65	65	61	61	61	61	57	57	61	61	57	57	57	57
13	65	65	61	61	61	61	57	57	61	61	57	57	57	57	53	53	81	81
14	61	61	57	57	61	61	57	57	57	57	53	53	81	81	65	65	61	61
15	61	61	57	57	57	57	53	53	81	81	65	65	61	61	61	61	57	57
16	57	57	53	53	81	81	65	65	61	61	61	61	57	57	61	61	57	57
17	81	81	65	65	61	61	61	61	57	57	61	61	57	57	57	57	53	53

4 The analysis of CLQ-algorithm

In this section, we will analyze the performance of CLQ-algorithm. In [2], Chen et al. proposed the CLQ-algorithm for computing the control tag of the backward network of a GSEN. Suppose a request is to send a message from port j (on the right-hand side) to port i (on the left-hand side). Then the input to CLQ-algorithm are i and j and the output of this algorithm is the backward control tag S for sending a message from j to i. CLQ-algorithm first computes the forward control tag T, which can be used to send a message from i to j; then, it converts T into S. The following is CLQ-algorithm; note that we assume $k = 2$.

BEGIN-of-CLQ-algorithm

Input: i on the left-hand side and j on the right-hand side of a bidirectional GSEN.

Output: The backward control tag S, which can be used to send a message from j to i.

1. Use P-algorithm to obtain a forward control tag T $(t_0t_1t_2\cdots t_{n-1}t_n)$.

2. Get the sequence R_ℓ $(0 \le \ell \le n)$ in the path based on tag T in forward direction by the following formulae:

$$R_\ell = \begin{cases} k \cdot i \bmod N' + t_0, & \ell = 0, \\ k \cdot (R_{\ell-1}) \bmod N' + t_\ell, & 1 \le \ell \le n. \end{cases} \tag{4}$$

3. Use the sequence R_ℓ and tag T to get tag S by the following formulae:

$$s_\ell = \begin{cases} \lfloor \frac{k \cdot i}{N'} \rfloor, & \ell = 0, \\ \lfloor \frac{k \cdot (R_{\ell-1})}{N'} \rfloor, & 1 \le \ell \le n. \end{cases} \tag{5}$$

END-of-CLQ-algorithm

We now analyze the performance of CLQ-algorithm. Suppose the two requests that are sent simultaneously are the (i, j)-request and the (i', j')-request. For each of the two requests, CLQ-algorithm first uses P-algorithm to obtain a forward control tag; then, it converts the forward control tag into a backward control tag. Let T and T' be the forward control tags obtained by P-algorithm for the (i,j)-request and the (i',j')-request, respectively. Also, let $\mathcal{P}$ and $\mathcal{Q}$ be the path corresponding to T and T', respectively. Since CLQ-algorithm fulfills the (i,j)-request by using $\mathcal{P}$ reversely and fulfills the (i',j')-request by using $\mathcal{Q}$ reversely, we have the following lemma.

Lemma 10. *The reverse path of $\mathcal{P}$ and the reverse path of $\mathcal{Q}$ has a link-conflict (node-conflict) if and only if $\mathcal{P}$ and $\mathcal{Q}$ have a link-conflict (node-conflict).*

Proof. This lemma follows from the above discussion. ∎

By Lemma 10, the analysis of P-algorithm can be used to obtain the analysis of CLQ-algorithm. In particular, for $N' = 4, 6, \cdots, 46$, Observations 1 to 7 and Tables 1 to 10 also hold for CLQ-algorithm.

5 The analysis of CL-algorithm

In this section, we will analyze the performance of CL-algorithm. In [3], Chen and Lou proposed the CL-algorithm for computing the control tag of the backward network of a GSEN. They showed that the backward network has a wonderful property: For each destination i, there are two backward control tags associated with it such that every source j can get to i by using one of the two tags.

In CL-algorithm, the switching elements are assumed to be of size $k \times k$. CL-algorithm is based on the following observations: At stage 0, only one switching element can get to i. At stage 1, exactly k switching elements can get to i and these switching elements are consecutive. At stage 2, exactly k^2 switching elements can get to i and these switching elements are consecutive. In general, at stage ℓ, $0 \le \ell \le n-1$, exactly k^ℓ switching elements can get to i and these switching elements are consecutive. At stage n (the last stage), all the switching elements can get to i. Since at stage ℓ the switching elements that can get to i are consecutive, CL-algorithm only stores the label of the first one; let C_ℓ denote the label. Chen and Lou have proven that $C_\ell = i \times k^\ell \bmod r$ and defined a critical value $v(i)$ associated with i to be $v(i) = C_n \times k$. The following is CL-algorithm; note that we assume $k = 2$.

BEGIN-of-CL-algorithm

Input: i on the left-hand side of a bidirectional GSEN.

Output: The critical value $v(i)$ and the two backward control tags $s_0\ s_1\ \cdots\ s_n$ and $s'_0\ s'_1\ \cdots\ s'_n$ associated with i (here s_ℓ and s'_ℓ are used at stage ℓ).

```
1. /* Compute $C_0, C_1, \cdots, C_n$. */
   for $\ell = 0$ to $n$ do $C_\ell \leftarrow (i \times k^\ell) \bmod r$;

2. /* Compute the critical value $v(i)$. */
   $v(i) \leftarrow C_n \times k$;

3. /* Compute $F_0, F_1, \cdots, F_n$. */
   if $(r - C_{n-1}) \times k \geq r$
   then
     begin
       for $\ell = 0$ to $n-1$ do $F_\ell \leftarrow 0$;
       $F_n \leftarrow 1$;
     end
   else
     for $\ell = 0$ to $n$ do if $C_\ell + k^\ell > r$ then $F_\ell \leftarrow 1$ else $F_\ell \leftarrow 0$;

4. /* Compute the tag $s'_0\ s'_1\ \cdots\ s'_n$. */
   $s'_0 \leftarrow \left\lfloor \frac{i}{r} \right\rfloor$;
   for $\ell = 1$ to $n$ do $s'_\ell \leftarrow \left\lfloor \frac{k \times C_{\ell-1}}{r} \right\rfloor$;

5. /* Compute the tag $s_0\ s_1\ \cdots\ s_n$. */
   for $\ell = 0$ to $n$ do $s_\ell \leftarrow (s'_\ell + F_\ell) \bmod k$;
```

END-of-CL-algorithm

Chen and Lou proved that:

Theorem 11. [3] *If $j < v(i)$, then j can get to i by using the control tag $s_0\ s_1\ \cdots\ s_n$; if $j \geq v(i)$, then j can get to i by using the control tag $s'_0\ s'_1\ \cdots\ s'_n$ (here s_ℓ and s'_ℓ are used at stage ℓ).*

We now analyze the performance of CL-algorithm. Suppose the two requests that are sent simultaneously are the (i, j)-request and the (i', j')-request. Also suppose the (i, j)-request is fulfilled by using the control tag S and the (i', j')-request, the control tag S'. Let $\mathcal{P}$ and $\mathcal{Q}$ be the paths corresponding to S and S', respectively. We run computer programs for $N' = 4, 6, \cdots, 46$ and find that for these N', the result of using $\mathcal{P}$ is identical to the result of using control tag T_1 for the (i, j)-request in the forward network; also, the result of using $\mathcal{Q}$ is identical to the result of using control tag T_1 for the (i', j')-request in the forward network. In particular, for $N' = 4, 6, \cdots, 46$, Observations 1 and 2 (when x is T_1T_1) and Tables 1 and 5 also hold for CL-algorithm.

6 Concluding remarks

A multistage interconnection network enables processors to send their messages concurrently. When two routing requests occur simultaneously, a link-conflict or a node-conflict may occur. In this thesis, we analyze the performance of three tag-based routing algorithms of GSENs: P-algorithm, CLQ-algorithm, and CL-algorithm. P-algorithm can be used in a GSEN and in the forward network of a GSEN; CLQ-algorithm and CL-algorithm can be used in the backward network of a GSEN. In this thesis, we focus on the case that *two* routing requests occur simultaneously and we consider the link-conflict and the node-conflict.

We have run computer programs for $N' = 4, 6, \cdots, 46$. From the computer output, the following observations have been obtained for P-algorithm and CLQ-algorithm:

- TotalLCF$(i, *, T_1T_1) >$ TotalLCF$(i, *, T_1T_2)$ whenever $N' \neq 2^{n+1}$.

- TotalNCF$(i, *, T_1T_1) >$ TotalNCF$(i, *, T_1T_2)$ whenever $N' \neq 2^{n+1}$ and $4|N'$.

- TotalNCF$(i, *, T_1T_1) <$ TotalNCF$(i, *, T_1T_2)$ whenever $N' \neq 2^{n+1}$ and $4 \not| N'$.

The above observations tell us that: When P-algorithm or CLQ-algorithm is used, two requests have a higher probability to have

- **a link-conflict** if $N' \neq 2^{n+1}$ and both requests are fulfilled by using the same type of control tags (that is, if both of them are fulfilled by using their T_1 or T_2 control tags);

- **a node-conflict** if $N' \neq 2^{n+1}$ and $4|N'$ and both requests are fulfilled by using the same type of control tags;

- **a node-conflict** if $N' \neq 2^{n+1}$ and $4 \not| N'$ and both requests are fulfilled by using the different types of control tags.

The following table has been obtained by Chen and Lou in [3].

the time required to	**CLQ-algorithm**	**CL-algorithm**
find a tag for a j to get to i	$O(n)$	$O(n)$
find the tags for every j to get to i	$O(N'n)$	$O(n)$
construct the routing table	$O(N'^2n)$	$O(N'n)$

This table tells us that: CL-algorithm is more efficient than CLQ-algorithm when we want to construct a routing table or when more than one control tag to the same destination needs to be found.

In this thesis, the following observations have been obtained for CL-algorithm: For $N' = 4, 6, \cdots, 46$, the result of using CL-algorithm is identical to the result of the T_1T_1-case. The above observation suggests that: When there are two requests and

- when link-conflict is considered, CLQ-algorithm beats CL-algorithm if $N' \neq 2^{n+1}$;

- when node-conflict is considered, CLQ-algorithm beats CL-algorithm if $N' \neq 2^{n+1}$ and $4|N'$;

- when node-conflict is considered, CL-algorithm beats CLQ-algorithm if $N' \neq 2^{n+1}$ and $4 \not| N'$.

The percentage of link-conflicts and node-conflicts for $N' = 4, 6, \cdots, 46$ are listed in Tables 9 and 10; we also show these data in Figures 3 and 4. In Table 9 (Table 10), T_1T_1 means that both the first and the second request are routed by using their T_1 control tags; T_1T_2 means that the first request is routed by using its T_1 control tag and the second request, its T_2 control tags; "arbitrary" means that only when all of the four types, T_1T_1, T_1T_2, T_2T_1, and T_2T_2, fail to contribute two link-disjoint (node-disjoint) routing paths, we will consider a conflict occurs.

Table 9: Percentage of link-conflicts.

N'	T_1T_1	T_1T_2	arbitrary
4	6.25	6.25	6.25
6	10.19	9.26	1.85
8	7.81	7.81	7.81
10	10.60	10.00	0.40
12	9.95	7.87	4.17
14	7.51	7.43	5.10
16	6.64	6.64	6.64
18	8.54	8.30	0.07
20	8.65	6.50	0.90
22	6.82	6.72	1.31
24	7.58	5.67	3.82
26	5.79	5.76	3.03
28	5.52	5.36	3.90
30	5.09	5.08	4.37
32	4.79	4.79	4.79
34	5.95	5.87	0.01
36	6.46	4.68	0.15
38	5.26	5.21	0.26
40	6.26	4.03	1.23
42	4.72	4.69	0.72
44	4.90	4.24	1.31
46	4.29	4.27	1.21

Table 10: Percentage of node-conflicts.

N'	T_1T_1	T_1T_2	arbitrary
4	31.25	31.25	31.25
6	32.41	36.11	32.41
8	26.56	26.56	26.56
10	26.60	33.00	20.20
12	24.77	23.84	16.44
14	21.50	22.08	19.75
16	19.14	19.14	19.14
18	18.96	25.14	11.01
20	19.85	19.25	9.65
22	17.04	19.37	11.63
24	17.77	15.68	11.98
26	15.08	15.85	12.26
28	14.27	14.19	11.86
30	13.39	13.47	12.56
32	12.60	12.60	12.60
34	12.46	16.82	5.79
36	13.86	13.62	5.39
38	11.85	14.42	5.93
40	13.46	11.31	5.71
42	11.11	12.69	6.32
44	11.21	11.11	5.71
46	10.37	11.39	6.74

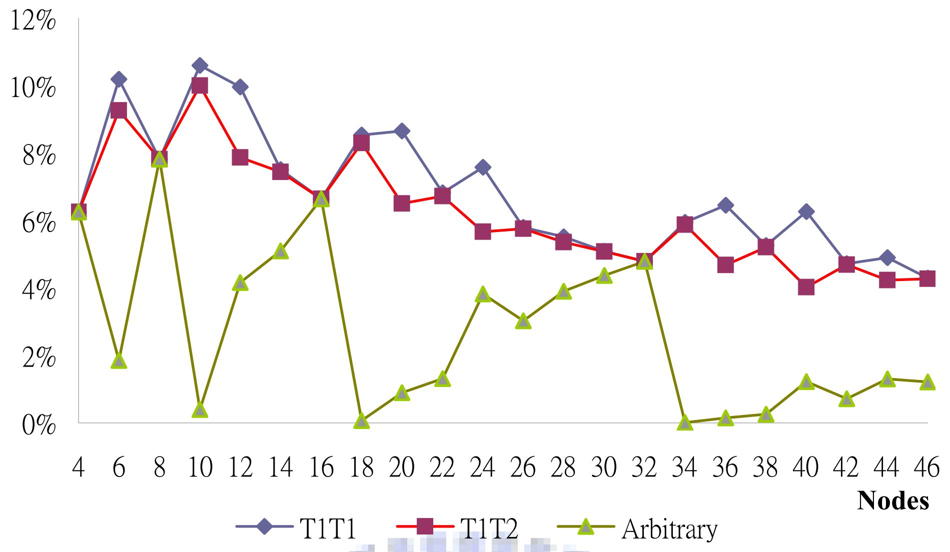

Figure 3: Percentage of link-conflicts.

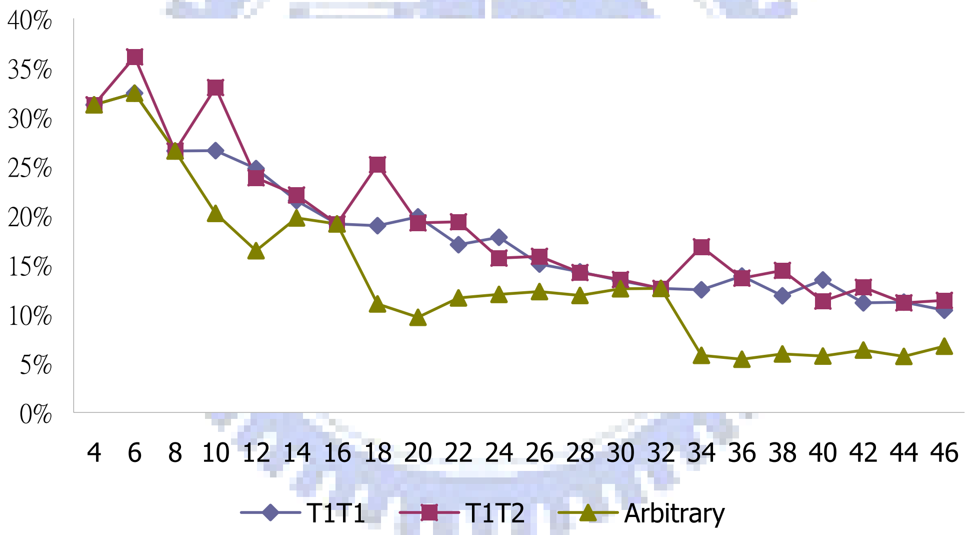

Figure 4: Percentage of node-conflicts.

References

[1] G. J. Chang, F. K. Hwang, and L. D. Tong, "Characterizing bit permutation networks," *Networks*, vol. 33, no. 4, pp. 261-267, 1999.

[2] Z. Chen, Z. J. Liu, and Z. L. Qiu, "Bidirectional shuffle-exchange network and tag-based routing algorithm," *IEEE Commun. Lett.*, vol. 7, no. 3, pp. 121-123, 2003.

[3] C. Y. Chen, J. K. Lou, "An efficient tag-based routing algorithm for the backward network of a bidirectional general shuffle-exchange network," *IEEE Commun. Lett.*, vol. 10, no. 4, pp. 296-298, 2006.

[4] M. Gerla, E. Leonardi, F. Neri, and P. Palanti, "Routing in the bidirectional shufflenet," *IEEE-ACM Trans. Netw.*, vol. 9, no. 1, pp. 91-103, 2001.

[5] F. K. Hwang, "The mathematical theory of nonblocking swithcing networks," *Series on Applied Mathematics,* vol. 15, 2004.

[6] C. P. Kuruskal, "A unified theory of interconnection network structure," *Theoretical Computer Science*, vol. 48, pp. 75-94, 1986.

[7] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, vol. 24, no. 12, pp. 1145-1155, 1975.

[8] V. W. Liu, C. Y. Chen, and R. B. Chen, "Optimal all-to-all personalized exchange in d-nary banyan multistage interconnection networks," *J. Comb. Optim.*. vol. 14, pp. 131-142, 2007.

[9] A. Massini, "All-to-all personalized communication on multistage interconnection networks," *Discrete Appl. Math.*, vol. 128, no. 2, pp. 435-446, 2003.

[10] K. Padmanabham, "Design and analysis of even-sized binary shuffle-exchange networks for multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 2, no. 4, pp. 385-397, 1991.

[11] C. Qiao and L. Zhou, "Scheduling switching element disjoint connections in stage-controlled photonic banyans," *IEEE Trans. Commun.*, vol. 47, no. 1, pp. 139-148, 1999.

[12] R. Ramaswami, "Multiwavelength lightwave networks for computer communication," *IEEE Commun. Mag.*, vol. 31, no. 2, pp. 78-88, 1993.

[13] Y. Yang, J. Wang, "All-to-all personalized exchange in banyan networks," *Proc. Parallel and Distributed Computing and Sysetems* (*PDCS'99*), Cambridge, MA, pp. 78-86, 1999.

[14] Y. Yang, J. Wang, "Optimal all-to-all personalized exchange in multistage networks," *Proc. Seventh International Conference on Parallel and Distributed Systems* (*ICPADS'00*), Iwale, Japan, 2000.

[15] Y. Yang, J. Wang, "Optimal all-to-all personalized exchange in self-routable multistage networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 3, pp. 261-274, 2000.

[16] Y. Yang, J. Wang, "Optimal all-to-all personalized exchange in a class of optical multistage networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 9, pp. 567-582, 2001.