Routing Permutations in Partitioned Optical Passive Stars Networks

Alessandro Mei [a] Romeo Rizzi [b]

[a] *Department of Computer Science, University of Rome "La Sapienza", Italy*

[b] *Department of Information and Communication Technology, University of Trento, Italy*

Abstract

It is shown that a Partitioned Optical Passive Stars (POPS) network with g groups and d processors per group can route any permutation among the $n = dg$ processors in one slot when $d = 1$ and $2\lceil d/g \rceil$ slots when $d > 1$. The number of slots used is optimal in the worst case, and is at most the double of the optimum for all permutations π such that $\pi(i) \neq i$, for all i.

Key words: Optical interconnections, passive stars, permutation routing.

1 Introduction

The Partitioned Optical Passive Star (POPS) network [1,3,2,5] is a SIMD interconnection network that uses multiple optical passive star (OPS) couplers. A $d \times d$ OPS coupler (see Figure 1) is an all-optical passive device which is capable of receiving an optical signal from one of its d sources and broadcast it to all of its d destinations. Being a passive all-optical technology, it benefits from a number of characteristics such as no opto-electronic conversion, high noise immunity, and low latency.

The number of processors of the network is denoted by n, and each processor has a distinct index in $\{0, \dots, n-1\}$. The n processors are partitioned into $g = n/d$ groups in such a way that processor i belongs to group $\text{group}(i) := \lfloor i/d \rfloor$. It is assumed that d divides n, consequently, each group consists of d processors. For each pair of groups $a, b \in \{0, \dots, g-1\}$, a coupler $c(b, a)$ is introduced which has all the processors of group a as sources and all the processors of group b as destinations. The number of couplers used is g^2. Such an architecture will be denoted by $\text{POPS}(d, g)$ (see Figure 2).

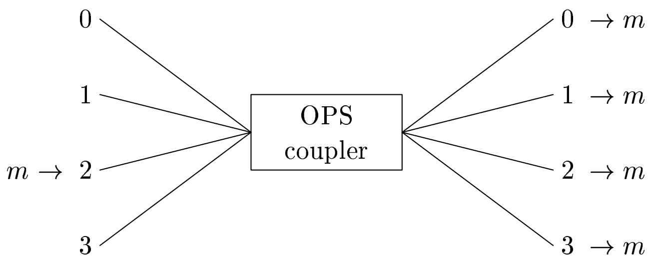

Fig. 1. A 4×4 Optical Passive Star (OPS) coupler.

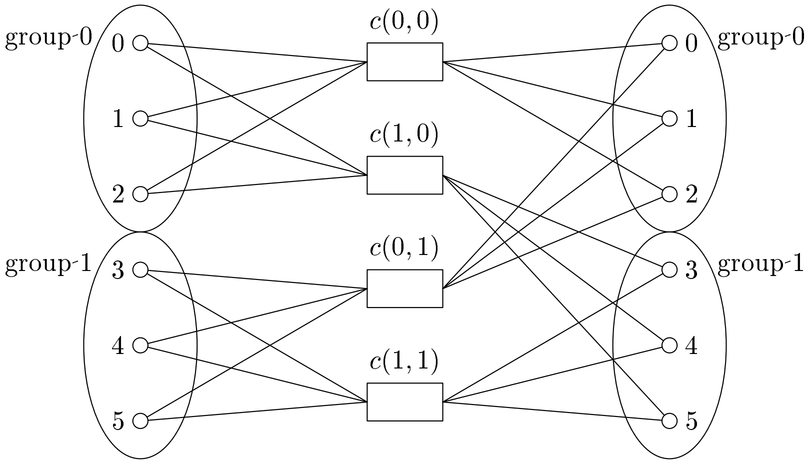

Fig. 2. A POPS$(3,2)$.

For all $i \in \{0,\ldots,n-1\}$, processor i has g transmitters which are connected to couplers $c(a,\text{group}(i))$, $a = 0,\ldots,g-1$. Similarly, processor i has g receivers connected to couplers $c(\text{group}(i),b)$, $b = 0,\ldots,g-1$. During a step of computation, each processor in parallel:

- Performs some local computations;
- sends a packet to a subset of its transmitters;
- receives a packet from one of its receivers.

In order to avoid conflicts, there shouldn't be any pair of processors sending a packet to the same coupler. The time needed to perform such a step is referred to as a *slot*.

One of the advantages of a POPS(d,g) network is that its diameter is 1. A packet can be sent from processor i to processor j, $i \neq j$, in one slot by using coupler $c(\text{group}(j),\text{group}(i))$. However, its bandwidth varies according to g. In a POPS$(n,1)$ network, only one packet can be sent through the single coupler per slot. On the other extreme, a POPS$(1,n)$ network is a highly expensive, fully interconnected optical network using n^2 OPS couplers.

A one-to-all communication pattern can also be performed in only one slot in the following way: Processor i (the speaker) sends the packet to all the couplers

$c(a, \text{group}(i))$, $a \in \{0, \ldots, g-1\}$, during the same slot all the processors j, $j \in \{0, \ldots, n-1\}$, can receive the packet through coupler $c(\text{group}(j), \text{group}(i))$.

The POPS network model has been used to develop a number of non trivial algorithms. Several common communication patterns are realized in [2]. Simulation algorithms for the mesh and hypercube interconnection networks can be found in [8]. Algorithms for data sum, prefix sum, consecutive sum, adjacent sum, and several data movement operations are also described in [8]. An algorithm for matrix multiplication is provided in [7]. These algorithms are based on sophisticated communication patterns, which have been investigated one by one, and shown to be routable on a POPS(d, g) network. However, most of these patterns belong to a more general class of permutation routing problems whose routability on the POPS network was not known in general. In this paper, we show that a POPS(d, g) network can efficiently route $n = dg$ packets arranged in the n processors according to any permutation, generalizing and unifying several known results appeared in the recent literature.

2 Definition of the Problem and Related Work

Let $\mathbb{N}_n := \{0, 1, \ldots, n-1\}$ denote the set of the first n natural numbers, and let π be a permutation of the set $\mathbb{N}_n$. A *permutation routing problem* consists of a set of n packets $p_0, \ldots, p_{n-1}$. Packet p_i is stored in the local memory of processor i, for all $i \in \mathbb{N}_n$, and has a desired destination $\pi(i)$. The problem is to route the packets to their destinations in as few slots as possible.

No general solution has been given for this problem on the POPS network. Efficient routings are known for a few particular permutations, which have been independently attacked, and most of them require one slot when $d = 1$ and $2\lceil d/g \rceil$ slots when $d > 1$. Here follow a few examples.

In [2], a characterization is given of the permutation routing problems that can be routed in a single slot. However, only a very restricted number of permutations fall in this class. Indeed, if π is such that two packets originating at the same group are to be routed to the same destination group, then one slot is obviously not enough to route π.

In [8], several permutation routing problems are considered in the context of the simulation of hypercube and mesh-connected computers on the POPS network. Assume that processor i of an $n = 2^D$ processor SIMD hypercube is mapped onto processor i of a POPS(d, g) network, $dg = n$. For every fixed b, $0 \leq b < D$, a primitive communication pattern is defined such that processor i sends a packet to processor $i^{(b)}$, where $i^{(b)}$ is the number whose binary representation differs from that of i only in bit b. Each of the D communication patterns defined is a permutation routing problem. Theorem 1 of [8] shows that all of them can be routed in one slot when $d = 1$ and $2\lceil d/g \rceil$ slots when $d > 1$.

The same result has been obtained when considering the problem of simulating an $N \times N$ SIMD mesh with wraparound, where data can be moved one processor up/down along the columns of the mesh, or right/left along the rows of the mesh. Again, assuming that processor (i, j) of the mesh is mapped onto processor $i + jN$ of a POPS(d, g) network ($dg = N^2$ and either d or g divides N), Theorem 2 of [8] shows that one slot when $d = 1$ and $2\lceil d/g \rceil$ slots when $d > 1$ are enough to route each of the four permutation routing problems.

The routability of other specific permutation routing problems is investigated in [7]. For example, a vector reversal (a permutation routing problem, where $\pi(i) = n - 1 - i, 0 \leq i < n$) is shown to be routable in one slot when $d = 1$ and $2\lceil d/g \rceil$ slots when $d > 1$ on a POPS(d, g) network, $dg = n$, which is optimal when g is even. To route a matrix transpose, conversely, $\lceil d/g \rceil$ is the optimal number of slots required.

Moreover, [7] considers BPC permutations. A BPC permutation is a rearrangement of the bits of the source processor index, while some or all of the bits can be complemented. Formally, assume that n is a power of 2, $n = 2^k$, and that the binary representation of i is $[i_{k-1} i_{k-2} \cdots i_0]_2$, the set of BPC permutations is the smallest set BPC closed under composition such that:

(1) $\pi(i) = \left[i_{\sigma(k-1)} i_{\sigma(k-2)} \cdots i_{\sigma(0)}\right]_2 \in$ BPC, for all σ permutation of $\mathbb{N}_k$;
(2) $\pi(i) = \left[i_{k-1} \cdots \overline{i_j} \cdots i_0\right]_2 \in$ BPC, for all j.

Again, [7] describes how BPC permutations can be routed in one slot when $d = 1$ and $2\lceil d/g \rceil$ slots when $d > 1$ on a POPS(d, g) network, $dg = n$.

In this paper we unify, generalize, and simplify the previously known results, by showing that a POPS(d, g) network, $dg = n$, can route *any* permutation in one slot when $d = 1$ and $2\lceil d/g \rceil$ slots when $d > 1$. This gives evidence of the versatility of the network. For example, a consequence of our Theorem 4 is that the simulation results for hypercube and mesh-connected computers shown in [8] do not depend on how the processors of the simulated architecture are mapped onto the processors of the POPS network, provided that it is a one-to-one mapping, which is somewhat surprising.

3 Routing Permutations in the POPS network

Assume the permutation routing problem defined by π on a POPS(d, g) network, $dg = n$, where π is a permutation of N_n. Our goal is to prove that π can be routed in one slot when $d = 1$ and $2\lceil d/g \rceil$ slots when $d > 1$.

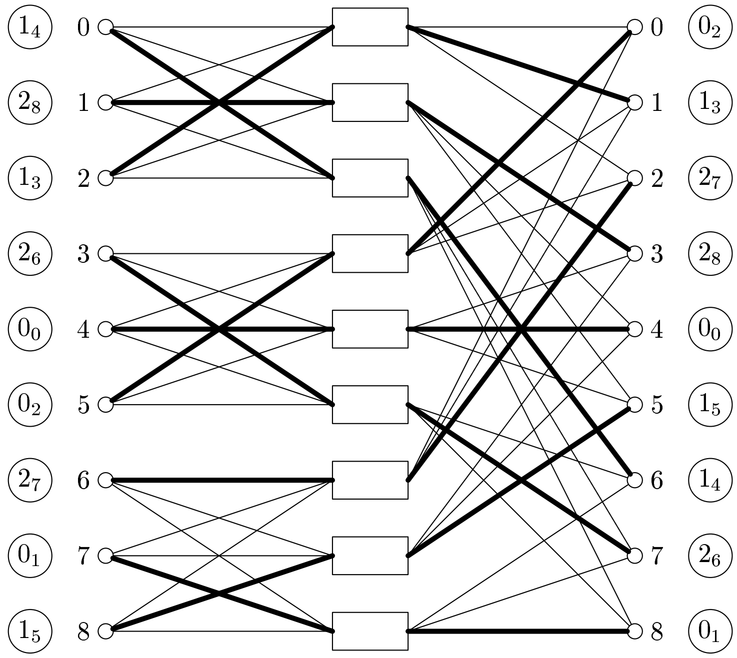

Fig. 3. Getting to a fair distribution on a $\mathrm{POPS}(3,3)$. Packets are drawn as circles next to their sources on the left. Inside each packet its destination x_y can be found, where y is the index of the destination processor, and x is its group. On the right, the intermediate destination of the packet as described in Section 3.1.

We start, for the ease of explanation, from the case $d = g = \sqrt{n}$. In this case, for most permutations one slot is not enough to route all the packets to destination. Take, as an example, the permutation shown in Figure 3. Packets starting from processor 4 and processor 5, both belonging to group 1, have the same group 0 as desired destination. If only one slot is allowed, there is an unavoidable conflict on coupler $\mathrm{c}(0,1)$. Hence, two slots are necessary to route π.

The above argument suggests a necessary and sufficient condition for a set of packets to be routable in one slot. We will say that m packets, each with a different destination, are arranged according to a *fair distribution* in a $\mathrm{POPS}(d,g)$ network if no two packets are stored in the same processor, and no two packets with the same destination group are stored in the same group. In this case, we will also say that the packets are *fairly distributed*. It is straightforward to see that a fairly distributed set of packets is routable in one slot. Indeed, no conflict occurs on any coupler.

Fact 1 *In a* $\mathrm{POPS}(d,g)$ *network, a fairly distributed set of m packets can be routed to destination in one slot.*

When $d = g = \sqrt{n}$, only a very small number of permutations can be routed in one slot. However, we will show that all of them can be routed in two slots. The idea is that one slot is always enough to move a set of n packets arranged according to π in such a way to become fairly distributed. Then, a second one routes all the packets to destination by Fact 1.

Next, in Subsection 3.1, we formalize the above intuition, and demonstrate our claim, properly generalized in order to deal with any value of d and g. Note that, for a set of packets to be fairly distributed, we don't really need to care about their processor destination. What we need is just to know what *group* destination each packet has. Thus, in Subsection 3.1 we can reduce our discussion to source groups and destination groups. d packets originate at each source group, and d packets have a specific destination group.

3.1 Permutation Routing: Getting to a Fair Distribution

A *list system* is a triple $(S,T,\mathcal{L})$, where S is a set of $n_1 := |S|$ *source nodes*, T is a set of $n_2 := |T|$ *target nodes*, and $\mathcal{L} : S \times \mathbb{N}_{\Delta_1} \mapsto S$ assigns a list L_s of $\Delta_1 \leq n_2$ not necessarily distinct elements from S to every source node $s \in S$. We also let $l(s,s')$ specify how many times the element $s' \in S$ appears into list L_s. A list system is called *proper* when n_2 divides $n_1\Delta_1$, and $\sum_{s\in S} l(s,s') = \Delta_1$ for every $s' \in S$.

Let $\Delta_2 := \frac{n_1\Delta_1}{n_2}$. A *fair distribution* for $(S,T,\mathcal{L})$ is an assignment $f : S \times \mathbb{N}_{\Delta_1} \mapsto T$ such that

$$|\{f(s,i) \mid i \in \mathbb{N}_{\Delta_1}\}| = \Delta_1 \text{ for every } s \in S; \tag{1}$$

$$|\{(s,i) \in S \times \mathbb{N}_{\Delta_1} \mid f(s,i) = t\}| = \Delta_2 \text{ for every } t \in T; \tag{2}$$

$$\begin{aligned}&\text{if } (s_1,i_1) \neq (s_2,i_2) \text{ and } \mathcal{L}(s_1,i_1) = \mathcal{L}(s_2,i_2), \text{then}\\ &f(s_1,i_1) \neq f(s_2,i_2), \text{ for every } s_1,s_2 \in S \text{ and}\\ &\text{every } i_1,i_2 \in \Delta_1.\end{aligned} \tag{3}$$

Theorem 2 *Every proper list system admits a fair distribution.*

PROOF. Let $S' := \{s' \mid s' \in S\}$. Consider the bipartite multigraph $G = (S,S';E)$, on node classes S and S', and having precisely $l(s,s')$ edges with one endnode in s and the other in s'. Clearly, for every $s \in S$, E contains precisely Δ_1 edges incident with s, namely the edges $\{s,\mathcal{L}(s,i)\}$ for $i \in \mathbb{N}_{\Delta_1}$. Moreover, for every $s' \in S'$, E contains precisely Δ_1 edges incident with s', since the list system is proper. The problem of finding a fair distribution for this proper list system can now be translated into the problem of finding an edge-coloring of G with n_2 ($\geq \Delta_1$ and such that n_2 divides $n_1\Delta_1$) colors and such that each color class has size precisely $\Delta_2 := \frac{n_1\Delta_1}{n_2}$.

Let V be a set of $n_1 - \Delta_2$ new nodes and $V' := \{v' \mid v \in V\}$. Let $H_1 = (V,S';F_1)$ be any bipartite $(n_2,n_2-\Delta_1)$-regular bipartite graph on node classes V and S'. Let $H_2 = (V',S;F_2)$ be any bipartite $(n_2,n_2-\Delta_1)$-regular bipartite graph on node classes V' and S. Consider the bipartite n_2-regular multigraph $\overline{G} = (S\cup V,S'\cup V';E\cup F_1\cup F_2)$. By König's theorem (1916, see e. g. Theorem 7.1.7. at page 276 of [10]), we can edge-color $\overline{G}$ with n_2 colors, that is, we can decompose $E\cup F_1\cup F_2$ into n_2

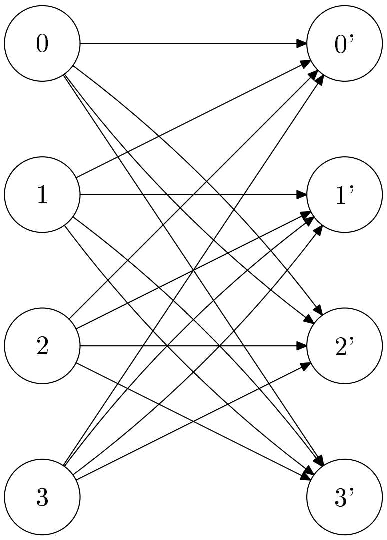

Fig. 4. Bipartite graph $\overline{G}$ generated, as described in the proof of Theorem 2, from proper list system $(\mathbb{N}_4, \mathbb{N}_4, \{\{2,2,0,3\},\{3,0,1,3\},\{1,3,2,0\},\{1,2,0,1\}\})$.

perfect matchings $M_0, \ldots, M_{n_2-1}$ of $\overline{G}$. We propose $M_0 \setminus F_1 \setminus F_2, \ldots, M_{n_2-1} \setminus F_1 \setminus F_2$ as the required edge-coloring of G. Indeed, $M_0 \setminus F_1 \setminus F_2, \ldots, M_{n_2-1} \setminus F_1 \setminus F_2$ is a decomposition of E into n_2 matchings of G and $|M_i \setminus F_1 \setminus F_2| = |M_i| - (|V| + |V'|) = (n_1 + |V|) - 2|V| = n_1 - |V| = n_1 - n_1 + \Delta_2 = \Delta_2$, for every $i = 0, \ldots, n_2 - 1$.

Remark 3 *The above proof is algorithmic. The computational bottleneck is in computing a* 1*-factorization of a bipartite* n_2*-regular multigraph on* $n := 4n_1 - 2\Delta_2$ *nodes and with* $m := nn_2$ *edges. This can be done in* $O(n_2 m)$ *as in [9] or in* $O(m \log n_2 + \frac{m}{n_2} \log \frac{m}{n_2} \log n_2)$ *as in [4] and in virtue of the algorithm described in [6].*

To help understand the construction of a fair distribution, consider the following list system: $(S, T, \mathcal{L}) = (\mathbb{N}_4, \mathbb{N}_4, \{\{2,2,0,3\},\{3,0,1,3\},\{1,3,2,0\},\{1,2,0,1\}\})$. This list system is proper, and, following the proof of Theorem 2, leads to the bipartite graph $\overline{G}$ shown in Figure 4. Graph $\overline{G}$ is 4-regular, therefore the set of its edges can be decomposed into four perfect matchings $M_0, \ldots, M_3$. Say that M_0 is equal to $\{(0,0'),(1,3'),(2,1'),(3,2')\}$. We can start building a possible fair distribution f by mapping all the list elements related to the edges in M_0 to the same element in T, say 0:

$$
\begin{aligned}
f = \{&\{*,*,0,*\},\\
&\{0,*,*,*\},\\
&\{0,*,*,*\},\\
&\{*,0,*,*\}\}.
\end{aligned}
$$

To complete the construction of f, map each list element related to an edge in M_i, $i = 1,2,3$, to i. As an example, say that $M_1 = \{(0,2'),(1,0'),(2,3'),(3,1')\}$, $M_2 = \{(0,2'),(1,3'),(2,0'),(3,1')\}$, and $M_3 = \{(0,3'),(1,1'),(2,2'),(3,0')\}$; a possible resulting fair distribution f is:

$$f = \{\{1,2,0,3\},\\ \{0,1,3,2\},\\ \{0,1,3,2\},\\ \{1,0,3,2\}\}.$$

It is easy to verify that f has all the properties in Equations 1, 2, and 3.

3.2 Permutation Routing: the Main Theorem

The following theorem describes our main result. Note that the routing found by Theorem 4 has the property that at each step of computation each processor stores exactly one packet.

Theorem 4 *A* POPS(d,g) *network can route any permutation* π *among the* $n = dg$ *processors using one slot when* $d = 1$ *and* $2\lceil d/g \rceil$ *slots when* $d > 1$.

PROOF. When $d = 1$, a POPS$(1,n)$ network is equivalent to an n processor clique, the network is fully interconnected, and the claim of the theorem is thus trivial.

Now, consider the case when $1 < d \leq g$. We will show that π can be routed in $2\lceil d/g \rceil = 2$ slots. Take the list system $(\mathbb{N}_g, \mathbb{N}_g, \mathcal{L})$, where $\mathcal{L} : \mathbb{N}_g \times \mathbb{N}_d \mapsto \mathbb{N}_g$ is such that $\mathcal{L}(h,i) = \text{group}(\pi(i+hd))$, $h \in \mathbb{N}_g, i \in \mathbb{N}_d$. The list system is proper, since π is a permutation, and g clearly divides gd. By Theorem 2, $(\mathbb{N}_g, \mathbb{N}_g, \mathcal{L})$ admits a fair distribution $f : \mathbb{N}_g \times \mathbb{N}_d \mapsto \mathbb{N}_g$. Consequently, f maps every pair (h,i) to an integer from $\mathbb{N}_g$ in such a way that:

$$|\{f(h,i) \mid i \in \mathbb{N}_d\}| = d \text{ for every } h \in \mathbb{N}_g; \tag{4}$$

$$|\{(h,i) \in \mathbb{N}_g \times \mathbb{N}_d \mid f(h,i) = j\}| = d \\ \text{for every } j \in \mathbb{N}_g; \tag{5}$$

$$\text{if } (h_1,i_1) \neq (h_2,i_2) \text{ and } \mathcal{L}(h_1,i_1) = \mathcal{L}(h_2,i_2), \text{ then} \\ f(h_1,i_1) \neq f(h_2,i_2), \text{ for every } h_1,h_2 \in \mathbb{N}_g \text{ and} \\ \text{every } i_1,i_2 \in \mathbb{N}_d. \tag{6}$$

Permutation π is routed in two slots. During the first slot, n packets are routed through n of the g^2 couplers of the POPS network, and, precisely, the packet originating at processor $i + hd$ is sent through coupler $c(f(h,i),h)$, $h \in \mathbb{N}_g, i \in \mathbb{N}_d$. No conflict can occur on any coupler by Equation (4). Moreover, exactly d packets arrive at group h by Equation (5), hence, it is easy to assign a distinct processor to read each of the incoming packets. After the first slot, the n packets are fairly distributed by Equation (6). Consequently, a second slot is enough to route all of them to destination by Fact 1.

Finally, consider the case when $d > g$. Take the list system $(\mathbb{N}_g, \mathbb{N}_d, \mathcal{L})$, where $\mathcal{L} : \mathbb{N}_g \times \mathbb{N}_d \mapsto \mathbb{N}_g$ is such that $\mathcal{L}(h,i) = \text{group}(\pi(i + hd))$, $h \in \mathbb{N}_g, i \in \mathbb{N}_d$. The list system is proper, since π is a permutation, and d clearly divides gd. By Theorem 2, $(\mathbb{N}_g, \mathbb{N}_d, \mathcal{L})$ admits a fair distribution $f : \mathbb{N}_g \times \mathbb{N}_d \mapsto \mathbb{N}_d$. Consequently, f maps every pair (h,i) to an integer from $\mathbb{N}_g$ in such a way that Equation (4), Equation (6), and the following Equation (7) hold:

$$|\{(h,i) \in \mathbb{N}_g \times \mathbb{N}_d \mid f(h,i) = j\}| = d \text{ for every } j \in \mathbb{N}_d. \tag{7}$$

Permutation π is routed in $\lceil d/g \rceil$ rounds. Each round k, $k = 0, \ldots, \lceil d/g \rceil - 1$, consists of two slots. During the first slot of all rounds but the last one, g^2 packets are routed through the g^2 couplers of the POPS network, and, precisely, the packet originating at processor $i + kg + hd$ is sent through coupler $c(f(h, i + kg), h)$, $h \in \mathbb{N}_g, i \in \mathbb{N}_g$. No conflict can occur on any coupler by Equation (4). Moreover, exactly g packets arrive at group h by Equation (7), hence, it is easy to assign a distinct processor (among the g which just sent a packet) to read each of the incoming packets. After the first slot, the g^2 packets which moved are fairly distributed by Equation (6). Consequently, a second slot is enough to route all of them to destination by Fact 1. The last round is exactly identical to the previous ones when g divides d. Otherwise, only $g(d \mod g)$ packets are routed in a similar way. After $\lceil d/g \rceil$ rounds all packets are correctly routed to destination.

The routing is completed after $\lceil d/g \rceil$ rounds, and each round consists of two slots. Consequently, π is routed using one slot when $d = 1$ and $2\lceil d/g \rceil$ slots when $d > 1$, as claimed.

The routing described by the previous theorem can be computed efficiently. The bottleneck consists in finding a fair distribution for the list system described by π, as in Theorem 2 and Remark 3. It is easy to see that this can be done in $O(g^3)$ or $O(g^2 \log g)$, when $1 < d \leq g$, and in $O(dn)$ or $O(n \log d)$ time, when $d > g$, by using the algorithms in [9] and [4,6], respectively.

3.3 Optimality

Theorem 4 is not far from optimality for almost all permutations. Indeed, if π is such that $\pi(i) \neq i$ for all i, then the routing found by Theorem 4 uses at most the double of the optimal number of slots.

Proposition 5 *If π is such that $\pi(i) \neq i$ for all i, then a* POPS(d,g) *network must use at least $\lceil d/g \rceil$ slots to route π.*

PROOF. Under the above assumptions, all packet destinations are different from the source. Hence, at least one slot is needed by each packet to reach the desired destination. Since a $\mathrm{POPS}(d,g)$ network can move at most g^2 packets per slot, $\lceil n/g^2 \rceil = \lceil d/g \rceil$ slots must be used to route all the packets.

Moreover, there exist permutations for which Theorem 4 is optimal. One example is vector reversal (when g is even), the proof can be found in [7]. A straightforward generalization of the proof in [7] shows that many other permutations have the same property.

Proposition 6 *If* π *is such that* $\mathrm{group}(i) \neq \mathrm{group}(\pi(i))$ *and*

$$\mathrm{group}(i) = \mathrm{group}(j) \Rightarrow \mathrm{group}(\pi(i)) = \mathrm{group}(\pi(j))$$

for all i and j, then a $\mathrm{POPS}(d,g)$ *network,* $dg = n$*, must use at least* $\lceil 2d/g \rceil$ *slots to route* π.

Finally, also when the assumption that $\mathrm{group}(i) \neq \mathrm{group}(\pi(i))$ is removed our algorithm gets very close to an optimal number of slots.

Proposition 7 *If* π *is such that* $\pi(i) \neq i$ *for all i and*

$$\mathrm{group}(i) = \mathrm{group}(j) \Rightarrow \mathrm{group}(\pi(i)) = \mathrm{group}(\pi(j))$$

for all i and j, then a $\mathrm{POPS}(d,g)$ *network,* $dg = n$*, must use at least* $\lceil 2d/(1+g) \rceil$ *slots to route* π.

PROOF. Suppose that a $\mathrm{POPS}(d,g)$ network can route π in t slots. If $t > d$, then it is easy to see that $t \geq \lceil 2d/(1+g) \rceil$. Hence, we can assume without loss of generality that $t \leq d$.

Since $\mathrm{group}(i) = \mathrm{group}(j) \Rightarrow \mathrm{group}(\pi(i)) = \mathrm{group}(\pi(j))$, at most t packets per group can be routed to destination in one slot only. All the other packets, at least $d-t$ per group, have to perform at least 2 hops to get to destination. Taking into account that a $\mathrm{POPS}(d,g)$ network can move at most g^2 packets per slot, then $tg^2 \geq gt + 2g(d-t)$, which implies that $t \geq \lceil 2d/(1+g) \rceil$.

4 Conclusion

A few papers appeared in the recent literature describing how data can be moved efficiently in a $\mathrm{POPS}(d,g)$ network. In particular, several permutation routing problems have been independently attacked in order to show they are routable in one slot when $d = 1$ and $2\lceil d/g \rceil$ slots when $d > 1$. With Theorem 4, we demonstrate that exactly the same result holds for any permutation π, and that the routing for π can be efficiently computed. Moreover, the number of slots used is optimal for a class of permutations, and at most twice of the number of slots required by any permutation π such that $\pi(i) \neq i$ for all i.

References

[1] D. Chiarulli, S. Levitan, R. G. Melhem, J. Teza, and G. Gravenstreter. Multiprocessor interconnection networks using partitioned optical passive star (POPS) topologies and distributed control. In *Proceedings First International Workshop on Massively Parallel Processing Using Optical Interconnections*, pages 70–80, 1994.

[2] G. Gravenstreter and R. G. Melhem. Realizing common communication patterns in partitioned optical passive stars networks. *IEEE Transactions on Computers*, 47(9):998–1013, September 1998.

[3] G. Gravenstreter, R. G. Melhem, D. Chiarulli, S. Levitan, and J. Teza. The partitioned optical passive star (POPS) topology. In *Proceedings Ninth International Parallel Processing Symposium*, pages 4–10, 1995.

[4] A. Kapoor and R. Rizzi. Edge-coloring bipartite graphs. *Journal of Algorithms*, 34(2):390–396, 2000.

[5] R. G. Melhem, G. Gravenstreter, D. Chiarulli, and S. Levitan. *The Communication Capabilities of Partitioned Optical Passive Star Networks*, pages 77–98. Kluwer Academics Publishers, 1998.

[6] R. Rizzi. Finding 1-factors in bipartite regular graphs, and edge-coloring bipartite graphs. *SIAM Journal on Discrete Mathematics*, 15(3):283–288, 2002.

[7] S. Sahni. Matrix multiplication and data routing using a partitioned optical passive stars network. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):720–728, July 2000.

[8] S. Sahni. The partitioned optical passive stars network: Simulations and fundamental operations. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):739–748, July 2000.

[9] A. Schrijver. Bipartite edge-colouring in $o(\delta m)$ time. *SIAM Journal on Computing*, 28(3):841–846, 1999.

[10] D. B. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ, 1996.