Journal of Universal Computer Science, vol. 11, no. 6 (2005), 944-961
submitted: 28/4/05, accepted: 25/6/05, appeared: 28/6/05 © J.UCS

On Complexity of Collective Communications on a Fat Cube Topology

Vladimír Kutálek
(Brno University of Technology, Czech Republic
kutalek@fit.vutbr.cz)

Václav Dvořák
(Brno University of Technology, Czech Republic
dvorak@fit.vutbr.cz)

Abstract: A recent renewed interest in hypercube interconnection network has been concentrated to the more scalable version known as a fat cube. The paper introduces several router models for fat nodes and uses them for cost comparison of both the hypercube and fat cube topologies. Analysis of time complexity of collective communications is done next and lower bounds on the number of communication steps are derived. Examples of particular communication algorithms on the 2D-fat cube topology with 8 processors are summarized and described in detail. The performed study shows that a large variety of fat cubes can provide much desired flexibility, trading cost for performance and manufacturability.

Keywords: Interconnection networks, fat cube topology, router architecture, collective communications
Categories: H.3.1, H.3.2, H.3.3, H.3.7, H.5.1

1 Introduction

One of the greatest challenges faced by designers of digital systems at present is optimizing the communication and interconnection between system components. As more and more processor cores and other large reusable components have been integrated on the single silicon die (MPSoCs, Multiprocessor Systems-on-Chips, [Jerraya, 05]), many of traditional multiprocessing techniques are modified or developed anew. The interconnection network, a fundamental component of every parallel system, and communication algorithms are no exceptions. E.g. buses, the former main means to connect the components, cannot scale to higher numbers of communication partners. Besides, long global wires become undesirable due to their low and unpredictable performance, what makes also the reuse of buses difficult. Buses are therefore being replaced by crossbars and by direct interconnection networks. For example, a high-speed crossbar switch is used in Niagara microprocessor chip [Geppert, 05] to connect 8 Sparc 4-way SMT microprocessor cores to a shared second-level cache. Among direct interconnection networks, traditional orthogonal topologies are also modified for performance-driven environments [Puente, 00]. Basically direct networks converge on the use of pipelined (wormhole) message transmission and source-based routing algorithms and the major difference between them are in topology.

Recently the research opened up in a Network on Chip (NoC) area, encompassing the interconnection/communication problem at all levels, from physical to the architectural to the OS and application levels [Jantsch, 03]. Packet switched networks suggested for NoC were up to now based either on 2D-mesh or on a fat tree topology [Jantsch, 03]. A novel on-chip communication architecture that can meet the performance requirements of network processors SoCs uses Octagon topology [Karim, 02]. However, aggressive on-chip communication demands not only of networking SoCs, but also SoCs in several other domains, can hardly be fulfilled by this network due to difficult scalability beyond 8 processors, suffering either wiring complexity or low performance in collective communications. In what follows we do not want to introduce a new topology, but rather look at possibilities of a derivative of the well-known topology: the fat cube. To the authors' knowledge, analysis of communication properties of this topology has not appeared in literature yet.

Binary hypercube topology is characterized by $P = 2^d$ nodes, naturally organized in d dimensions, where d is also the node degree. The worst-case distance between two nodes, network diameter D, is logarithmic, $D = d = \log P$. The hypercube topology is node and edge symmetric, what simplifies the design of parallel algorithms tremendously. Computation can start in any node and the source code remains the same. Optimal algorithms for collective communication operations exist in almost all communication models. This is why the hypercube topology is commonly considered the best topology there is from the algorithmic and communication viewpoint. The hypercube topology can simulate efficiently almost any other topology, too. The only drawback is its non-constant (logarithmic) degree $d = \log P$ and consequently a high number of communication channels and only partial scalability, as the number of nodes P is restricted to powers of 2.

Topologies derived from the binary hypercube, such as cube-connected cycles and wrap-around or ordinary butterflies [Dally, 04] eliminate the drawback of non-constant node degree. They are constructed by expanding the hypercube vertices into cycles or linear arrays and have a small constant degree and the logarithmic diameter as before. The bisection width $2^d = P/d$ is slightly worse than the value P for hypercube and so is the scalability, since the number of processors is $P = d2^d$, i.e. only 8, 24, 64, etc.

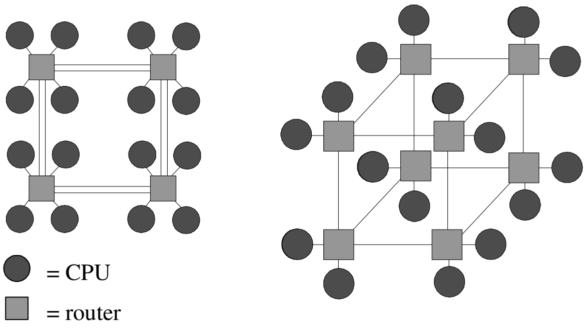
Another useful alternative is much better scalable topology called a "fat cube". The vertices of the hypercube are again expanded, but now into sets of processors connected by the crossbar switch inside the router. Scalability is improved since the node can contain any number of processors, $P = m2^d$ P, $m = 1, 2, 3, \ldots$ The node degree grows more slowly than in hypercube, $d = \log (P/m)$ and the bisection width can be adjusted by multiple links between nodes. Due to these favorable features has the fat cube topology been recently used e.g. in commercial DSM NUMA machine Origin 3000 (SGI). Also Opteron processors produced by AMD have hyperlinks on the chip ready for fat cube connection [Keltcher, 03]. Fat nodes (8 CPUs per node) have been used in Swiss-T1 cluster with K-ring network [Gabrielyan, 04]. The fat cube topology is also expected to appear in future networking systems for MPSoCs, because its mapping into 2-D space is easier than in the case of the "thin" hypercube.

Whereas the properties of the ordinary hypercube are well known, no theoretical results are available for fat cubes. The paper analyzes, for the first time, the complexity of several collective communication patterns on a fat cube network. The rest of the paper is the original research done by authors and it is organized as follows: In Section 2 we define the fat cube topology and four distinct router models. Section 3 presents the details of hardware cost calculation. Then we look at complexity of collective communications on the fat cube in Section 4. Section 5 is a case study involving all the important collective communications on several models of the 8-processor fat cube. Finally, Section 6 concludes this paper.

2 Fat cube and router architecture

Let us recall notation introduced above and establish some new notions related to the fat cube topology. Drawings of two instances of this topology are shown in Figure 1. We use the following parameters (the logarithm has an implicit base 2):

- d – dimensionality of the fat cube/hypercube, it is equal to the number of external input or output ports per router
- d'– dimensionality of the hypercube with $P = m2^d$ vertices, $d' = \lceil \log P \rceil = d + \lceil \log m \rceil$ (log is the binary logarithm)
- D – network diameter
- m – number of processors per fat node, an integer greater than 1
- P – processor count $P = m2^d$ (the fat cube), $P = 2^d$ (the hypercube)
- f – multiplicity of external links
- L – the number of external links in a fat cube network $L = fd2^{d-1}$
- $C(n,k) = n!/[(n-k)!k!]$



a) b)

Figure 1: Examples of fat cube networks:
a) $P = 16, m = 4, d = 2, f = 2$ b) $P = 16, m = 2, d = 3, f = 1$

The design of communication algorithms depends strongly on the model used to describe the parameters of the underlying communication hardware. These models have to address key characteristics of interconnection networks, such as switching technique, channel type, message combining capability and a router model. The possible options in communication architecture are:

1. SF | WH | CS | VCT – store-and-forward, wormhole, circuit, and virtual-cut-through switching techniques
2. HD | FD | S – half-duplex, full-duplex, simplex link type
3. NC | C – non-combining/ combining model capable or not, combining or extracting partial messages with negligible overhead
4. one-port (1) | all-port (*) | all-output -port (b) | d-port (d) router model.

The router model for fat nodes deserves some explanation, because it is a certain generalization of the router model used in connection with thin nodes. In the simplest case processors are connected to the router by a single link as in Fig. 1. This so called one-port model ("1") allows each of m or less processors to send a message either outside to a remote processor or to the local processor inside the same node, Fig. 2. All-output-port model ("b") is one-port model with possible broadcast facility to all output ports (external and internal). The broadcasting processor could simultaneously receive one message from outside, too. Model "b" will be treated separately only for one-to-all broadcast communication. In other communication patterns its behavior is identical to 1-port model. In d-port model ("d") each processor can send up to d distinct messages simultaneously, either outside or locally. In the last all-port model ("*"), each processor can simultaneously send up to ($d + m - 1$) distinct messages to all d dimensions and to all $m - 1$ remaining local processors. This model is rather of theoretical interest only because of a high router cost. The traditional hypercube may use only three special cases of the above router models defined by $m = 1$ and $f = 1$. Under these conditions models "d" and "*" become identical and are known as the all-port model.

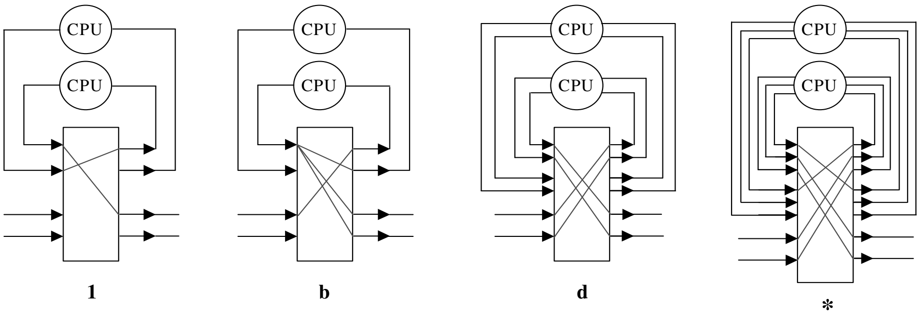


Figure 2: Router models for fat nodes ($m = 2, d = 2, P = 8, f = 1$)
*1) one-port model b) all-output-port model d) d-port model *) all-port model*

3 Cost of a fat cube network

The cost of the interconnection network has two components: the external links cost C_L and the router cost C_R. If we disregard manufacturability, the external link cost C_L can be taken simply as the number of these links $C_L = L = fd2^{d-1}$. The router cost given mainly by the cost of $a \times b$ crossbar switch with a input ports and b output ports is commonly taken as ab; in our analysis routers have square crossbars with $a = b$.

Let us compare the fat cube cost *C* and hypercube cost C' and let us find under which condition is the fat cube network cheaper. If both the networks have the same number of processors *P* = *P*', then

$$P = m2^d = P' = 2^{d'} \;\; or \;\; d' = d + \log m \tag{1}$$

The lower link cost $C_L \le C_L'$ of the fat cube

$$C_L = fd2^{d-1} \le C_L^{'} d'2^{d'-1} = d'm2^{d-1} \tag{2}$$

implies $fd \le md'$, what holds true because dimensionality of the fat cube is always lower than that of a hypercube with *P* nodes and because mostly $f \le m$.

The cost C_R of all routers together depends on the type of the port model. Table 1 compares the total router cost C_R and C_R', the product of input and output port counts, $p_{\text{in}} = p_{\text{out}}$. Of course, we are interested especially in fat cubes with some cost advantage, i.e. when $C_R \le C_R'$. By making use of relation (1) we can transform the condition of a lower cost into inequalities involving parameters *m*, *f* and *d*:

$$(1) \quad (m + df)^2 \le m(1 + d + \log m)^2 \tag{3}$$

$$(d) \quad d^2(m + f)^2 \le 4m(d + \log m)^2 \tag{4}$$

$$(*) \quad [d(m + f) + m(m-1)]^2 \le 4m(d + \log m)^2 \tag{5}$$

Table 2 shows some numerically obtained solutions of inequalities (3) – (5) for *f* = 1 and 2.

	Fat cube		Hypercube ($m = f = 1$)	
Model	$p_{\text{in}} = p_{\text{out}}$	Router cost (C_R)	$p_{\text{in}} = p_{\text{out}}$	C_R'
1	$(m+df)$	$2^d(m+df)^2$	$(1+d')$	$2^{d'}(1+d')^2$
d	$d(m+f)$	$2^d d^2(m+f)^2$	$2d'$	$2^{d'} 4 d'^2$
*	$d(m+f)+ m(m-1)$	$2^d(d(m+f)+m^2-m)^2$	$2d'$	$2^{d'} 4 d'^2$

Table 1: Total router cost in fat cube (C_R) and hypercube (C_R') topology

$f = 1$	1	d	*
$m = 2$	$\forall d$	$d \le 16$	$d \le 4$
$m = 4$	$\forall d$	$d \le 8$	*never*
$m = 8$	$\forall d$	$d \le 5$	*never*

$f = 2$	1	d	*
$m = 2$	$d = 1$	$d \le 2$	*never*
$m = 4$	$\forall d$	$d \le 6$	*never*
$m = 8$	$\forall d$	$d \le 3$	*never*

Table 2: Conditions ensuring that a fat cube be cheaper than the hypercube

For example, both fat cube networks at Fig. 1 (model 1) are cheaper then hypercubes (model 1) with the same number of processors *P*. Now the question is

CC	SF hypercube		WH hypercube	
	1-port	all-port	1-port	all-port
OAB	$\log P\ (= d)$	$D\ (= d)$	$\log P\ (= d)$	$\lceil \log_{d+1} P \rceil = \lceil d/\log(d+1) \rceil$
AAB	$P - 1$	$\lceil (P-1)/d \rceil$	$P - 1$	$\lceil (P-1)/d \rceil$
OAS	$P - 1$	$\lceil (P-1)/d \rceil$	$P - 1$	$\lceil (P-1)/d \rceil$
AAS	$d\,P/2$	$P/2$	$P - 1$	$P/2$

Table 3: CCs on a hypercube, lower bounds τ_{CC} on time complexity

what will be the impact of this lower hardware cost, if any, on communication performance. We will therefore investigate the performance of collective communications on a fat cube in the next section.

4 Complexity of collective communications on a fat cube

Collective communications (CCs) are frequently used in all parallel algorithms. In a phase parallel model [Hwang, 98] a parallel program is executed as a sequence of phases of three types, parallelism phase (process management), local computation, and interaction phase (communication, synchronization or aggregation). Collective communications are most important and if their overhead is excessive, performance degrades rapidly with the processor count. When we refer to „collective communications", we will assume communications involving *all* processors. Seven types of such collective communications are: OAB (One-to-All Broadcast), OAS (One-to-All Scatter), AOG (All-to-One Gather), AOR (All-to-one reduce), AAB (All-to-All Broadcast), AAR (All-to-all Reduce) and AAS (All-to-All Scatter), [Duato, 03]. Since complexities of some communications are similar (AOG ~ OAS, AOR ~ OAB, AAR ~ AAB), we will focus only on 4 basic types (OAB, OAS, AAB, AAS). Each communication may be investigated with all possible model options, what gives too many distinct cases to explore. Therefore only the most important of them will be analyzed.

In the rest of the paper we assume that the communication proceeds in synchronized steps. In one step of CC, a set of simultaneous packet transfers take place only between *adjacent* nodes (SF networks) or along complete disjoint paths between source-destination node pairs (WH networks). Complexity of collective communication will be determined in terms of the number of these communication steps τ(CC, G); if network graph G is clear from the context, we will omit its symbol G. This figure of merit does not take into account the message length or its variations from one step to another. Before analyzing communications on a fat cube, let us review the lower bounds on number of steps τ_{CC} in a hypercube network, Table 3. Lower bounds for all CCs on the SF hypercube are reachable by known optimal algorithms. However, an asymptotically optimal algorithm for OAB is not known on the all-port WH hypercube; the lower bound τ_{OAB} can be reached only for $d \le 6$ by the double-tree algorithm. Other known algorithms are nearly optimal (e.g. algorithm Ho-Kao, [Duato, 03]).

In the following subsections we want to generalize the above results for the fat cube topology with restriction to non-combining SF models with FD links.

4.1 One-to-all broadcast (OAB) on a SF fat cube

This CC is not influenced by the type of the links (HD / FD) or message (non)combining. Since just one message propagates in the network, we will consider only simple links ($f = 1$), even though multiple links could help in some way. On the 1-port fat cube we have the same lower bound as on the hypercube:

Theorem 1. Complexity of OAB on the 1-port fat cube is lower bounded by $\tau_{\mathrm{OAB}} = \lceil \log P \rceil$. This lower bound can be reached for all f, d, and m.

Proof. The theorem holds for a hypercube; it is well known that by making use of the binomial spanning tree [Duato, 03] we can reach all nodes in $d = \log P$ steps. The fat cube with $P = m2^d$ processors requires additional $\lceil \log m \rceil$ steps to inform local processors inside nodes. This intra-node communication may be overlapped with inter-node communication except the last node. The multiple external links have no influence on this, so that $d + \lceil \log m \rceil = \lceil \log P \rceil$ steps will always be needed, q.e.d.

For example in 1-port, 12-CPU fat cube network ($d = 2$, $m = 3$) we need 4 steps, the same number as for 16-CPU fat cube or hypercube.

Better OAB performance than in the hypercube can be obtained with more complex routers (d, b, *) in the fat cube network. In general, complexity of OAB on a k-port network defined by graph G is given by the larger of two values: D and $\lceil \log_{k+1} P \rceil$, $\tau_{\mathrm{OAB}} = \max\ [D, \lceil \log_{k+1} P \rceil\ ]$. For b- and d-port router models there are d external ports available for OAB and either component may dominate; diameter $D = d$ dominates over $\lceil \log_{d+1} P \rceil$ if

$$m \le \left[(d+1)/2\right]^d . \qquad (6)$$

This condition is satisfied in the grey area in Tab. 4 where is the lower bound better than in the equivalent hypercube (with the same processor count P) because the diameter of the fat cube $D = d$ is always lower than that of the equivalent hypercube.

	b- and d-model			*-model		
d / m	2	4	8	2	4	8
1	2	3	4	2	3	4
2	2*	4	5	2	2	2
3	3*	3	3	3	3	3
4	4*	4*	4	4	4	4

Table 4: Lower bound $\tau_{OAB} = \max\ [D, \lceil \log_{k+1} P \rceil\ ]$ as a function of d and m; $\tau_{OAB} = D$ in grey cells, asterisks denote values reachable according to Theorem 2.

If we consider *-model, there are $d+m-1$ ports available for OAB and diameter D dominates $\lceil \log_{d+1} P \rceil$ if $(d+m)^d \geq m2^d$ or equivalently if $d > 1$ and m is arbitrary, see Table 4.

The lower bound $\tau_{\mathrm{OAB}} = D = d$ on the 1-port fat cube with simple edges ($f = 1$) can also be reached in models "d" and "*" if $d \geq m$; otherwise we need additional steps to complete local OAB communication within certain nodes as given by Theorem 2:

Theorem 2. Complexity of OAB on NC SF fat cube measured by the number of communication steps is

d) $\tau^{\mathrm{OAB}} = d + \lceil \log_{d+1} \lceil m/d \rceil \rceil$

*, b) $\tau^{\mathrm{OAB}} = d$ (if $m \leq d$) or $d+1$ (if $m > d$).

Proof. First we shall prove

Lemma 1. The number of links between adjacent levels i and $i + 1$ of a broadcast tree in the d-dimensional fat cube network is $C(d-1, i)$.

Proof. The statement is valid for $i = 0$, see Figure 3. Assuming that there are $dC(d-1, i-1)$ links coming to the level i, we should prove that there are $dC(d-1, i)$ links coming to level $i+1$. Indeed, subtracting incoming links from all links incident with nodes at level i, we get

$$d\binom{d}{i} - d\binom{d-1}{d-i} = d\frac{d!}{(d-i)!i!} - \frac{d(d-1)!}{(i-1)!(d-i)!} = d\frac{(d-1)!}{(d-i)!}\left[\frac{d}{i!} - \frac{1}{(i-1)!}\right] =$$

$$= d\frac{(d-1)!}{(d-i)!}\left[\frac{d-i}{i!}\right] = d\binom{d-1}{i},$$

q.e.d.

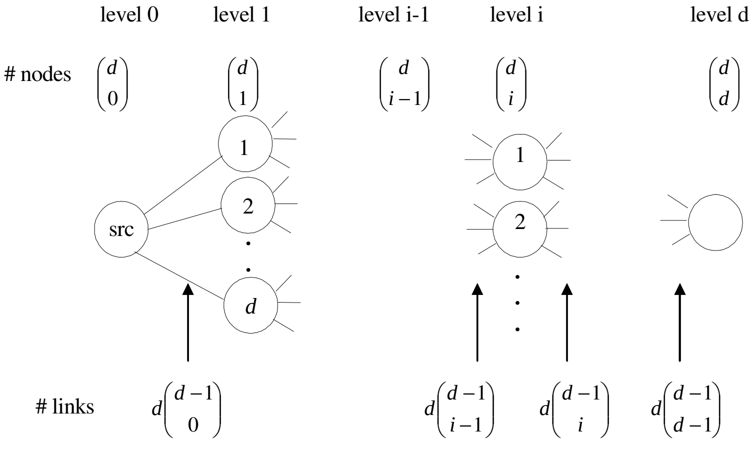

level 0
level 1
level i-1
level i
level d
# nodes
$\binom{d}{0}$
$\binom{d}{1}$
$\binom{d}{i-1}$
$\binom{d}{i}$
$\binom{d}{d}$
src
1
2
d
1
2
# links
$d\binom{d-1}{0}$
$d\binom{d-1}{i-1}$
$d\binom{d-1}{i}$
$d\binom{d-1}{d-1}$

Figure 3: Number of nodes and links in a broadcast tree of a fat cube network

To prove Theorem 2, let us start with step 1, when the source processor at level 0 sends a message along all $dC(d-1, 0) = d$ external links to all $C(d, 1)$ level-1 nodes. The source processor will inform remaining $m-1$ local processors in one more step (step 2) since $d > m-1$. In step 2 all informed level-1 processors will propagate the message to all $C(d, 2)$ level-2 nodes. Since, according to Lemma 1, there are $dC(d-1, 1)$ external lines from level 1 to level 2, i.e. 2-times more than the number of level-2 nodes, two processors in each node can be informed simultaneously. Again, remaining local processors will be informed in the next step 3, since $d > m-2$.

Similarly, informed processors at level $(i-1)$ will propagate the message to $dC(d-1, i-1)$ processors at level i, what gives $dC(d-1, i-1)/C(d, i) = i$ newly informed processors in each node. From a certain level i on, it will hold true $i \geq m$, so that all m processors in each node, starting at level i, will be informed simultaneously. In the worst case it will happen just at the last level if $i = d = m$. However, if we will have $i = d < m$, some additional steps will be needed. Having m-d processors in each node still uninformed, each of d informed processors can distribute the message within the subset of $\lceil m/d \rceil$ local processors in $\lceil \log_{d+1} \lceil m/d \rceil \rceil$ steps (d-port model) or in 1 step (*-model, because $d+m-1 > m-d$). Adding up d steps of inter-node communication (where simple links ($f = 1$) are sufficient) and the above extra steps of intra-node communication, we get the desired result.

Finally, model-b router is designed for fast OAB and if the message enters a node, it is distributed to all $d + m - 1$ output ports in the same step, regardless the relation between m and d. We therefore have, even with simple links, $\tau^{\mathrm{OAB}} = d + 1$, q.e.d.

Theorem 2 provides a remarkable result, because it says that at least for OAB we can get in some cases better performance (asterisks in Table 4) and in many cases also a lower cost (see Table 2) than with the equivalent hypercube.

4.2 All-to-all broadcast (AAB) on a SF fat cube

The lower bound τ_{AAB} is easily found from the number of messages that each processor has to *receive* (all receive simultaneously) and the number of its input ports as

$$\tau_{\mathrm{AAB}} = P - 1,\ \lceil (P-1)/d \rceil,\ \text{and}\ \lceil (P-1)/(d+m-1) \rceil \tag{7}$$

for "1", "d", and "*" router models, respectively. Optimal AAB algorithms for a hypercube matching the lower bounds are based on a Hamiltonian cycle (1-port model) and on so called time-arc-disjoint spanning trees – TADTs (all-port and *-models). All processors can use such broadcast trees synchronously with no conflicts. SF and WH switching give equal results, HD links can simulate FD links with slowdown two or sometimes better. The following Theorem 3 establishes complexity of AAB, namely upper bounds τ^{AAB} in case that we do AAB among nodes first and then inside the nodes. As we will see later, due to a possible partial overlap of both the inter- and intra-node communications in models d and *, lower bounds τ_{AAB} can be reached under certain conditions.

Theorem 3. Complexity of AAB on the NC SF fat cube measured by the number of communication steps is

1) $\tau^{\mathrm{AAB}} = \tau_{\mathrm{AAB}} = P-1$
d) $\tau^{\mathrm{AAB}} = \lceil (P-m)/\min(fd, md) \rceil + \lceil (m-1)/d \rceil P/m$
*) $\tau^{\mathrm{AAB}} = \lceil (P-m)/\min(fd, m(d+m-1) \rceil + P/m$.

Proof.

1) We can use cyclic rotation of messages along the ring formed by the Hamiltonian cycle, *m* processors in the node are incorporated into that cycle. In the first step all *P* processors are just sending their message along the cycle and in following *P*–2 cycles they keep receiving and re-sending other messages. Multiple links cannot make it faster, because processors are connected to the router with a simple link.

d) Using a generic TADT rooted in every node we will perform AAB among *nodes*. Each node broadcasts "super-messages" consisting of *m* distinct messages to (2^d-1) remaining nodes. In each "super-step", *m* messages stored in *m* node processors are transferred between adjacent nodes. There are (2^d-1) super-messages destined for one node, *fd* incoming links to a node from all dimensions and *md* input links to node processors. Therefore not more than min (*fd, md*) distinct messages can arrive to one receiver node simultaneously. Each super-message is thus sent in not less than $\lceil m/\min(fd, md) \rceil$ steps. At the end will each processor have *P*/*m* distinct messages (including its own original message) to share with other local processors. As the local AAB among *m* nodes can be done on the router crossbar as *m*–1 permutations, *d* permutations at a time, the result is

$$\tau^{\mathrm{AAB}} = (2^d-1)\lceil m/\min(fd, md) \rceil + \lceil (m-1)/d \rceil P/m = \\ \lceil (P-m)/\min(fd, md)) \rceil + \lceil (m-1)/d \rceil P/m, \qquad (8)$$

q.e.d.

*) The same reasoning as in d) except that there are now *m*(*d*+*m*–1) input links to a node. Local AAB, *m*–1 permutations, and *d* messages to external links can be combined into one step as each processor is connected to the crossbar by *d*+*m*–1 links. By taking *x* = min(*fd*, *m*(*d*+*m*–1)) we have

$$\tau^{AAB} = \lceil (P-m)/x \rceil + \lceil (m-1)/(d+m-1) \rceil P/m = \\ = \lceil (P-m)/x \rceil + P/m, \qquad (9)$$

q.e.d.

Provided that we have a practical model "d" with *f* < *m*, then *md*–*fd* ports are free during inter-node communication and can be used for broadcasting messages within the node. As there are (*P*–*m*)/(*fd*) steps of inter-node communication, $(\tilde{P}-m)(m\tilde{-}f)/f$ out of total $m[((\tilde{2}^d-1)+1)(\tilde{m}-1)] = P(m\tilde{-}1)$ internal pairwise communications can be hidden. Remaining $P(m-1)-(P-m)(m-f)/f$ pairwise communications can be done, *md* of them at a time, on *md* ports. With the previous inter-node communication it will require

$$\tau^{AAB} = \left\lceil \frac{P-m}{fd} \right\rceil + \left\lceil \frac{P(m-1)}{md} - \frac{P-m}{fd}\left(1-\frac{f}{m}\right) \right\rceil = \left\lceil \frac{P-1}{d} \right\rceil = \tau_{AAB} \qquad (10)$$

steps.

For model "*" with $fd < m(d+\tilde{m}-1)$ we can, similarly as under case d), overlap inter- and intra-node communications with the result

$$\tau^{AAB} = \left\lceil \frac{P-m}{fd} \right\rceil + \left\lceil \frac{P(m-1)}{m(d+m-1)} - \frac{P-m}{fd}\left(1-\frac{fd}{m(d+m-1)}\right) \right\rceil = \left\lceil \frac{P-1}{d+m-1} \right\rceil = \tau_{AAB} \qquad (11)$$

Contrary to OAB, combining is relevant to the complexity of AAB. There is a straightforward approach (Gather – Scatter) to *combining* AAB on the fat cube: one representative processor in each node gathers messages from all local peers and then AAB takes place among these representative processors with combined messages. At the end the representatives extract and distribute individual messages to local peers. We will not analyze complexity in detail, but interestingly, combining AAB can sometimes be faster on the fat cube than on the hypercube, [Kutalek, 05].

4.3 One-to-all scatter (OAS) on a SF fat cube

This CC has similar complexity as AAB in many models, the lower bound τ_{OAS} is easily found from the number (P – 1) of distinct messages that each processor has to now *send* (all processors send simultaneously) and the number of its input ports as

$$\tau_{\mathrm{OAS}} = P-1,\ \lceil (P-1)/d \rceil, \text{ and } \lceil (P-1)/(d+m-1) \rceil \qquad (12)$$

for 1, d, and * router models, respectively. Optimal OAS algorithms for a hypercube matching the lower bounds are based on a Hamiltonian cycle (1-port model) and again on time-arc-disjoint spanning trees TADTs (models d and *). Regardless of whether we have SF or WH switching, an optimal algorithm requires a broadcast tree with sub-trees of approximately equal size. TADTs do not fulfil this requirement and must be slightly modified. The construction of such trees is known and will not be repeated here. The generic TADT tree can be rooted in any source processor and using FF strategy, messages are pipelined within the sub-trees. HD or FD switching do not influence τ_{OAS}, rather the number of distinct messages that can be sent by the source processor in one step is important. In the fat cube topology we perform OAS among nodes first, then OAS inside nodes. Theorem 4 gives related upper bounds τ^{OAS}; for $m = f = 1$ we get the lower bounds for the hypercube as a special case.

Theorem 4. Complexity of OAS on NC SF fat cube measured by the number of communication steps is

1) $\tau^{\mathrm{OAS}} = \tau_{\mathrm{OAS}} = P-1$

d) $\tau^{\mathrm{OAS}} = \lceil (P-m)/\min(fd, md) \rceil + \lceil (m-1)/d \rceil$

*) $\tau^{\mathrm{OAS}} = \lceil (P-m-d)/\min(fd, m(d+m-1)) \rceil + 1$

Proof.

1) We can use cyclic rotation of messages along the ring formed by the Hamiltonian cycle, going through all processors of one node and then to the following node and through all its processors, etc. We use the farthest node first strategy (FF), sending the message to the most remote processor first and going on with messages for processors closer and closer to the source. We cannot use more than $f = 1$ external link, because each processor has only one internal link and both the external and internal links are connected in the Hamiltonian cycle. Therefore in the (P – 1)-th step all processors will get their messages, q.e.d.

d) The FF strategy is now used in TADT for OAS among nodes. There are $(2^d - 1)$ super-messages from source node destined for other nodes. In each super-step, m messages are transferred between adjacent nodes. There are fd outgoing links from a node in all dimensions and md output links from processors, so not more than min (fd, md) distinct messages can be sent by source node simultaneously. Each super-message is thus sent in $\lceil m/\min(fd, md)\rceil$ steps. Local OAS in the source node requires $\lceil (m-1)/d \rceil$ steps, because the source processor can emit d messages at a time. Altogether

$$\tau^{\mathrm{OAS}} = \lceil m/\min(fd, md)\rceil (2^d - 1) + \lceil (m-1)/d \rceil =$$
$$\lceil (P-m)/\min(fd, md)\rceil + \lceil (m-1)/d \rceil,$$

q.e.d. For simple links ($f = 1$) this bound comes to $\tau^{\mathrm{OAS}} = \lceil (P-1)/d \rceil = \tau_{\mathrm{OAS}}$.

*) The same reasoning as in d), only different number of links. The source processor can emit all $m-1$ local messages together with d external messages simultaneously in one step as it has $d + m - 1$ output ports. Therefore if $x = \min(fd, m(d+m-1))$, then

$$\tau^{\mathrm{OAS}} = \lceil (P-m-d)/x \rceil + \lceil (d+m-1)/(d+m-1) \rceil = \lceil (P-m-d)/x \rceil + 1, \text{ q.e.d.}$$

Let us note that for simple links ($f = 1$) this bound can be simplified to

$$\tau^{\mathrm{OAS}} = \lceil (P-m-d)/d \rceil + 1 = \lceil (P-m)/d \rceil.$$

4.4 AAS on a SF fat cube

The lower bounds τ_{AAS} will be determined later. Let us recall only that the optimal AAS algorithm for the all-port hypercube matching the lower bound $\tau_{\mathrm{AAS}} = P/2$ (see Table 3) runs according to a schedule that specifies triplets (step number, dimension, relative address RA of source and destination processors) in a way that avoids conflicts; an example is at Fig. 7. Algorithms exist for a construction of such schedules and resulting scheduling tables [Edelman, 91] and prove the elegance of hypercube topology: in d-port model all links are used in both directions in all steps! AAS lower bounds for HD and FD links differ again by factor of two (model NC). Theorem 5 concerns FD links and establishes complexity of AAS, namely upper bounds τ^{AAS} in case that we do AAS among nodes first and then inside the nodes. In some cases can these bounds be further improved by overlapping both inter- and intra-node communications.

Theorem 5. Complexity of AAS on NC SF fat cube measured by the number of communication steps is

1) $\tau^{\mathrm{AAS}} = \lceil Pmd/[2\min(m, fd)] \rceil + (m-1)$

d) $\tau^{\mathrm{AAS}} = \lceil Pmd/[2\min(md, fd)] \rceil + \lceil (m-1)/d \rceil$

*) $\tau^{\mathrm{AAS}} = \lceil Pmd/[2\min(m(d+m-1), fd)] \rceil + 1$

Proof.

1) We can visualize AAS as a superposition of m-to-P scatter communications by all nodes, in which each processor in the node sends P-m distinct messages outside and m-1 messages inside the node. The block of m^2 messages (a super-message) from

the source node (m source CPUs in one node, each of them sending m messages to a destination node) may stop over in intermediate nodes on the path to a destination node. Super-messages re-sent from intermediate nodes are to be taken as new super-messages and k of such super-messages will be generated on the path from level 0 to level k of the broadcast tree. Counting up all such super-messages for all destinations at all levels of a broadcast tree we get the total count x

$$x=\sum_{k=0}^{d}k\binom{d}{k}=0\binom{d}{0}+\sum_{k=1}^{d}\frac{kd(d-1)!}{(d-k)!k(k-1)!}=d\sum_{k=1}^{d}\binom{d-1}{k-1}=d2^{d-1}. \tag{13}$$

There will be m^2-times more simple external messages. Each node (model 1) is able to receive (or send) up to min(m, fd) external messages in one step, so that inter-node communication requires $\lceil m^2d2^{d-1}/\min(m, fd)\rceil = \lceil Pmd/[2\min(m, fd)]\rceil$ steps. The intra-node AAS among m processors can be implemented on the router crossbar as $m-1$ permutations in $m-1$ steps, and together we get the desired result on 1-port model, q.e.d.

For $m \le fd$ the above result can be simplified to

$$\tau^{\mathrm{AAS}} = dP/2+(m-1). \tag{14}$$

Not sooner than after $dP/2$ steps will be m ports free for the intra-node AAS. Eqn. (14) is thus also a lower bound since no shorter solution exists. If on the other hand $m > fd$, we get

$$\tau^{\mathrm{AAS}} = \lceil Pm/(2f)\rceil+(m-1). \tag{15}$$

In this case a possibility exists to overlap the intra-and inter-node communication. For full overlap [Kutalek, 05]

$$\tau^{\mathrm{AAS}} = \tau_{\mathrm{AAS}} = \lceil Pm/(2f)\rceil. \tag{16}$$

d) The proof goes on along the same lines with a bottleneck of

min (# internal links, # external links) = min (md, fd)

external messages in one step. AAS among local processors, m-1 permutations on the crossbar, can be done at a rate d permutations in one step, so that $\lceil (m-1)/d\rceil$ steps are needed, q.e.d. Simplified bounds are:

$$\text{if } m\le f\text{: } \tau^{\mathrm{AAS}} = \tau_{\mathrm{AAS}} = P/2+\lceil (m-1)/d\rceil \text{ and}$$
$$\text{if } f\le m\text{: } \tau^{\mathrm{AAS}} = \lceil Pm/(2f)\rceil+\lceil (m-1)/d\rceil. \tag{17}$$

Overlapping both global and local AAS is again possible in the latter case [Kutalek, 05].

*) The same reasoning as above for model d. Since every processor has now additional m–1 links above d-port model, m–1 permutations can now be performed in one step, since $\lceil (m-1)/(d+m-1)\rceil$ = 1, q.e.d.

If $f = m$, we get $\tau^{\mathrm{AAS}} = P/2 + 1$, but because in this case m–1 links to/from each processor are free during global AAS, we can combine 1 step of local AAS with any step of global AAS without interference between them, so that $\tau^{\mathrm{AAS}} = \tau_{\mathrm{AAS}} = P/2$.

Combining models can also be considered; combining messages in a super-step can be done as in the case of AAB, using Gather – Scatter algorithm. Again, combining AAS can sometimes be faster on the fat cube than on the hypercube, [Kutalek, 05].

5 Examples of collective communication on the 8-processor, 2D-fat cube

Here we will demonstrate communication algorithms on the small fat cube with the following parameters: $d = m = 2$, $P = 8$, $f = 1$, non-combining nodes, full duplex links, store and forward switching.

5.1 One-to-all broadcast

According to Theorems 1 and 2, OAB will take not more than 3, 2, 2, and 2 steps on 1, b, d, and * router models. Routing schedules with these models are shown in Figure 4. Router model "b" behaves like model "*" only in OAB communication, in other communications like model "1". Whereas 3 OAB steps are always needed in 8-processor hypercube, 2 steps will do in the fat cube topology (model d, b, and *). Routing follows exactly the algorithm described in proof of Theorem 2. Only in the 1-port model does the routing proceed according to the spanning binomial tree with 1+1+2+4 processors informed in 3 steps. Doubling the number of informed processors keeps going the same way on the interconnections as on the router crossbar (if m >2).

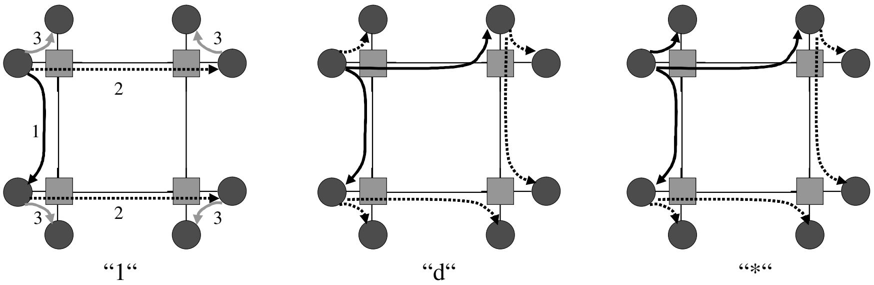
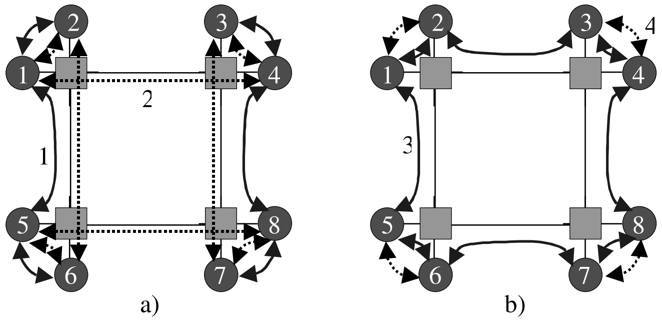


Figure 4: OAB communication on the SF fat cube

5.2 All-to-all broadcast

Theorem 3 states that we are able to complete AAB in 7 steps with all 1, d, and * router models, but if we manage to overlap inter- and intra-node communication, the lower bounds for d and * routers, i.e. 4 and 3 steps, can be reached. (From now on we do not mention model "b" explicitly, as it behaves like model "1"). The case of 1-port router model is easy when we use the Hamiltonian path. In case of d-port model, the optimal algorithm with the full overlap of the global and local AAB is shown at Figure 5, reaching the lower bound $\tau_{AAB} = 4$ steps. . The path of every message from source to destination processors, divided into 4 steps, is described in Table 5.


2
3
1
4
2
1
5
8
6
7
4
2
3
1
4
3
5
8
6
7

a) b)

Figure 5: AAB communication on the d-port SF fat cube
a) steps 1 and 2 b) steps 3 and 4

	destination processors			
message	step 1	step 2	step 3	step 4
1	→ 2, 5	→ 6, 8	→ 7, 4	→ 3
2	→ 1, 3	→ 4, 7	→ 8, 6	→ 5
3	→ 4, 2	→ 1, 6	→ 5, 7	→ 8
4	→ 3, 8	→ 7,5	→ 6, 1	→ 2
5	→ 6, 1	→ 2, 4	→ 3, 8	→ 7
6	→ 5, 7	→ 8, 3	→ 4, 2	→ 1
7	→ 8, 6	→ 5, 2	→ 1, 3	→ 4
8	→ 7, 4	→ 3, 1	→ 2, 5	→ 6

Table 5: 4 steps of the AAB communication schedule

5.3 One-to-all scatter

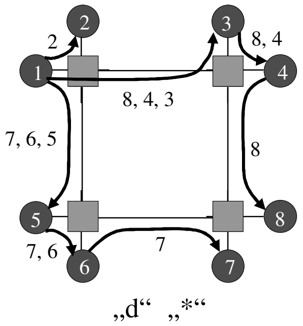
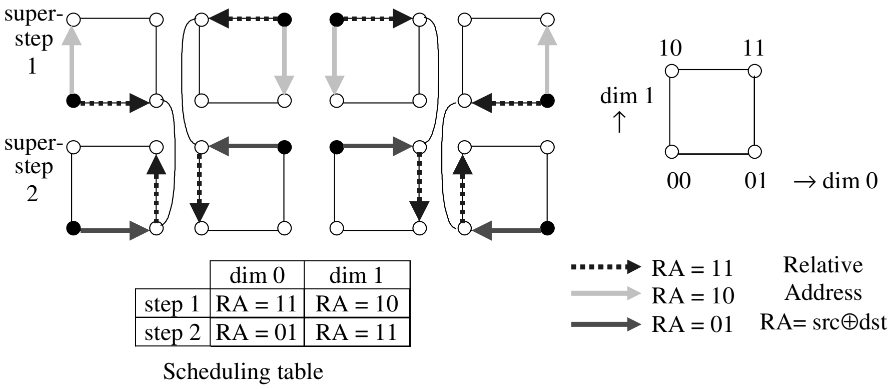

2
3
8, 4
1
4
8, 4, 3
7, 6, 5
8
5
8
7, 6
7
6
7
„d“ „*“

Figure 6: Farthest First message pipelining in OAS on the sample fat cube

The upper bounds given by Theorem 4 match the ideal lower bounds 7, 4, 3 in our running example for 1, d, and * router models, respectively. OAS in 7 steps with 1-port model is easy using the Hamiltonian cycle. Two other models (d and *) are more interesting and the sequences of messages on various links are depicted in Fig. 6. In model "d" the source keeps sending 2 messages into two sub-trees for 3 steps and 1 more step is for OAS inside the source node. This extra step can be combined with any step in model "*", so that Fig. 6 is sufficient for the both models.

5.4 All-to-all scatter

According to Theorem 5, we should be able to complete AAS on our example fat cube surprisingly with all types of routers in the same number of 9 steps. We shall start with the scheduling table for an all-port 2D-hypercube, Fig.7. AAS among nodes is scheduled in 2 super-steps. Considering now $m^2 = 4$ messages in a super-message, there will be 4 steps in each super-step. AAS within nodes, in our fat cube only exchange of messages between two processors, can be done in 1 extra step (1-port model) or can be combined with any of previous 8 steps (d-port model and even more so *-model), because only one processor port is busy during inter-node communication. Thus 8 steps will do for *- and d-port model, but 9 steps will be needed for 1-port model. Multiple links would improve performance further.



	dim 0	dim 1
step 1	RA = 11	RA = 10
step 2	RA = 01	RA = 11

Scheduling table

Figure 7: AAS communication on d-port SF 2D-hypercube and a scheduling table

6 Results and conclusions

Summary of CC complexities for various models of our sample fat cube and hypercube networks is in Table 6. All models of the SF NC fat cube with $m = 2$, $f = 1$ and $d = 2$ are cheaper then the 8-processor hypercube (see Table 2) and their communication performance is the same or better in OAB and almost the same in OAS and AAB. The AAS communication is 25% faster on 1-port models, but twice as slower as on the hypercube on models "d" and "b".

Model	NC/C		NC models					
	OAB		OAS		AAB		AAS	
	HC	FC	HC	FC	HC	FC	HC	FC
1	3	3	7	7*	7	7*	12	9*
b	3	2	7	7	7	7	12	9
d	3	2	3	4*	3	4*	4	8
*	3	2	3	3*	3	4	4	8

Table 6: Comparison of τ_{CC} for SF hypercube and fat SF cube networks with P=8 processors. (HC = hypercube, FC = fat cube (m=2, d=2, f=1), asterisks denote reaching the lower bound)

The above results concern only one particular fat cube network, but theorems derived earlier are suitable for comparison of other configurations as well. Generally we can make the following conclusions:

1. Performance in OAB is the same in 1-port router models of fat cubes and hypercubes.
2. OAB can be done faster on models b, d, and * thanks to the hardware branching capability built in the router (model b) or more processor ports for concurrent communications (models d and *).
3. Performance in OAS and AAB are similar as in hypercube topology, but they could run faster if multiple links are provided.
4. Router models with more ports have no advantage in AAS over models 1 and b, because utilization of multiple processor ports is very low.
5. Performance in AAS is given by the bisection width, which is only a half of the value in the hypercube. The same performance as in the hypercube can be obtained when multiple links are used.
6. Router models "*" have almost no advantage over models "d", because the amount of intra-node communication is much less than that of inter-node communication. Some overlap of these communications can be realized in AAB and OAS.
7. If the hardware cost is a limiting factor, then a suitable fat cube can be found which is cheaper than the equivalent hypercube with the same number of processors and with not much (if any) performance degradation.
8. The number of processors *P* in the fat cube configuration is not limited to powers of 2, but a power of 2 can be multiplied by an integer *m*. This may be more straightforward scaling than a partial hypercube.

The future research should address WH fat cubes and other network topologies with fat nodes and links. Also other communication patterns should be studied, such as multicast and m-to-n broadcast or scatter. Also combining node models are of interest; partial results for SF switching have been presented in [Kutalek, 05]. The role of combining models for WH switching should still be clarified. The research in the above directions could help optimize communication architectures for application-specific multiprocessor systems on chip, [Dvorak, 03].

Acknowledgement

This research has been carried out under the financial support of the research grant “Network Architectures of Embedded Systems Networks”, GA102/05/0467, Grant Agency of Czech Republic, 2005-2007.

References

[Jerraya, 05] A.A.Jerraya, W. Wolf: Microprocessor Systems-on-Chips. Elsevier Inc., 2005, ISBN 0-12385-251-X.

[Geppert, 05] L. Geppert: Sun′s Big Splash. IEEE Spectrum, Vol.42, No.1, Jan. 2005, p.50-54.

[Puente, 00] V. Puente, et al.: Improving Parallel System Performance by Changing the Arrangement of the Network Links. Proc. of the International Conference on Supercomputing, May 2000, p.44-53.

[Jantsch, 03] A. Jantsch, H. Tenhunen: Networks on Chip. Kluwer Academic Publ., 2003, ISBN 1-4020-7392-5

[Karim, 02] F. Karim, A. Nguyen: An Interconnect Architecture for Networking Systems on Chips. IEEE Micro, Sept. – Oct. 2002, pp. 36-45.

[Keltcher, 03] C.N. Keltcher, et al.: The AMD Opteron Processor for Multiprocessor Servers. IEEE Micro, March/April 2003, pp.66 – 76.

[Gabrielyan, 04] E. Gabrielyan, R.D. Hersch: Efficient Liquid Schedule Search Strategies for Collective Communications. Proc. of ICON 2004 - 12th IEEE International Conference on Networks, Singapore, Vol. 2, November 16-19, 2004, pp 760-766.

[Hwang, 98] K. Hwang, Z. Xu: Scalable Parallel Computing: Technology, Architecture, Programming. McGraw Hill, 1998.

[Kutalek, 05] V. Kutalek: Performance modeling and optimization of application-specific multi-processor systems. Ph.D. thesis, Faculty of information technology, Brno University of Technology, 2005.

[Duato, 03] J. Duato, S. Yalamanchili, L. Ni: Interconnection Networks – An Engineering Approach. Morgan Kaufman Publishers, 2003, ISBN 1-55860-852-4.

[Dvorak, 03] V. Dvorak: Communication Architectures for Application-Specific Multiprocessor Systems (on a Chip). Proc. of the 11th International Conference on Software, Telecommunications and Computer Networks SoftCOM 2003, Split, HR, FESB, 2003, p. 778-782.

[Dally, 04] W. Dally, B. Towles: Principles and Practices of Interconnection Networks. The Morgan Kaufmann Series in Computer Architecture and Design, Morgan Kaufman Publishers, 2004, ISBN: 0-12200-751-4

[Edelman, 91] A. Edelman: Optimal matrix transposition and bit reversal on hypercubes: All-to-all personalized communication. Journal of Parallel and Distributed Computing, Vol. 11, p.328-333, 1991.