Message Routing on Irregular 2D-Meshes and Tori

Thomas M. Stricker[1]

January 15, 1991

CMU-CS-91-109

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Wormhole message routing is supported by the communication hardware of several distributed memory machines. This particular method of message routing has numerous advantages but creates the problem of a *routing deadlock*. When long messages compete for the same channels in the network, some messages will be blocked until the the first message is fully consumed by the processor at the destination of the message. A deadlock occurs if a set of messages mutually blocks, and no message can progress towards its destination. Most deadlock free routing schemes previously known are designed to work on regular binary hypercubes. Regular hypercubes and meshes are just a special case of networks. However, these routing schemes do not provide enough flexibility to deal with irregular 2-D-tori and with attached auxiliary cells, which can be found on many newer parallel systems.

To handle irregular topologies elegantly, a simple proof is necessary to verify the router code. The new proof given in this report is carried out directly on the network graph. It is constructive in the sense that it reveals the design options to deal with irregularities and shows how additional flexibility can be used to achieve better load balancing.

Based on the modified routing model, a set of deadlock free router functions relevant to the iWarp system configurations are described and proven to be correct.

[1]Address of the author: School of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213, Telephone (412) 268 2580, E-mail: tomstr@cs.cmu.edu. Support: The research was supported in part by the Defense Advanced Research Project Agency, US Department of Defense, monitored by the Space and Naval Warfare Systems Command under Contract N00039-87-C-0251, and in part by the Office of Naval Research under Contracts N00014-87-K-0385 and N00014-87-K-0533.

Keywords: wormhole message routing, channel dependecy, routing deadlocks, virtual channels, iWarp system architecture

1 Introduction

A unit to handle data transfers between different processors is part of most modern distributed memory machines; this unit is either integrated with the main processor or can be implemented off chip as a coprocessor. We call this unit *communication agent*. A typical function performed by the communication agent is forwarding messages without any participation of the main processing unit in that cell. The communication agent of the iWarp component [iWarp90] supports wormhole message routing as well as other schemes. Wormhole routing is one of the fastest forwarding techniques currently known.

In wormhole routing usually only one link is used to connect adjacent cells. A physical link can be used by a single channel or be shared among multiple channels. The communication agent contains switching hardware to distribute the incoming messages to the appropriate output channels and control the flow of messages. This switching system needs to restrict the message traffic in some specific way to avoid a routing deadlock. Routing deadlock will eventually occur when long messages compete for access to one of the links, and some messages have to wait until the first message is fully consumed by the processor at the destination of a message. In a deadlock situation none of the mutually blocking messages can progress towards its destination, since there is a circular dependency of blocked messages.

The message traffic used in numerous parallel algorithms is very regular and does usually not expose a deadlock. However, most distributed memory systems include system software resident in every cell (i.e. a runtime or operating-system kernel). Communication related to maintenance (like loading, debugging and profiling) happens in addition to the data transfers of the parallel algorithms. A communication system capable of handling this kind of message traffic can not make any assumption about regular message patterns and must exclude deadlock by virtue of its routing method.

Previous work has addressed the problem mainly on k-ary hypercube topologies [DS87]. Typical iWarp systems are two-dimensional $n \times m$ tori with partially connected *auxiliary cells* for input/output operations. Auxiliary cells handle communication with input/output subsystems of the parallel machines. Examples of I/O subsystems include: workstations and host computers, video framebuffers, disk-farms, or fiber-optical high speed networks. Auxiliary cells will play a major role in the iWarp systems since the architecture is well suited for image- and signal-processing in real-time. Therefore, the basic runtime system router must be able to deal well with the auxiliary cells as well as with the regular mesh.

Auxiliary cells can be inserted into the surface of the mesh or into the torus wrap-around links, which we call *the border of the mesh*. An example of an iWarp configuration is shown in Figure 1, labeled with all the terms used in this report.

2 Routing deadlock

An iWarp cell can be linked to up to four neighbors with high speed physical links. The communication hardware integrated in the iWarp component supports several logical channels over one physical link. This allows general message passing communication to coexist with very specific systolic communication. For most parts of this report we assume that the message passing uses one set of logical channels corresponding exactly to the physical buses that link cells, and we will refer to them simply as *channels*.

The wormhole message routing technique does not allow transient messages to be buffered in local memory. Whenever a message tries to move into an occupied channel, its header is blocked and the tail fills the channel buffers along its path. The route of a message is determined by the sender and contained in the message header. There is no rerouting.

A closer look to wormhole message passing reveals that even if every processor eventually consumes all

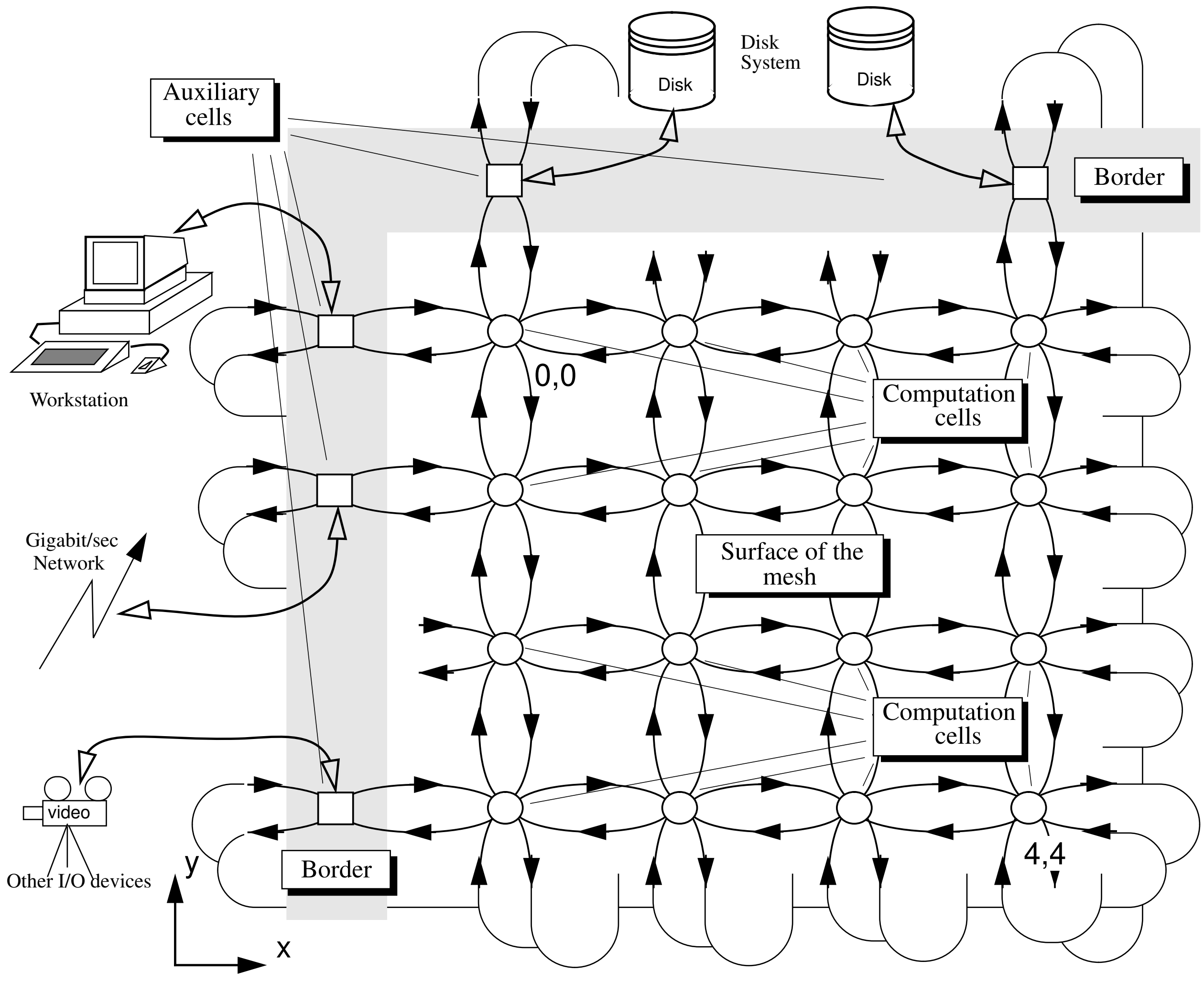

Figure 1: Terms used to describe a typical iWarp torus.

messages addressed to that cell, some messages can mutually block within the forwarding communication agents.

2.1 Deadlock configurations

A set of messages are considered to be deadlocked if they block a cyclic path with a routing dependency.

Figure 2 illustrates blocking routes in a two-dimensional torus. Although there are waiting messages on the channel in Figure 2, they are not blocked in a deadlock since message “A” can resume its travel as soon message “B” is consumed by its destination processor and “B” is removed from the channels. 3 illustrates a set of waiting messages that are actually blocked in a routing deadlocked. The messages “A”,“B”,“C”,“D” mutually block each other in a cycle and none of them can advance towards its destination.

A more formal definition of the term *deadlock configuration* is established in [DS87] based on the terminology of channel dependency graphs.

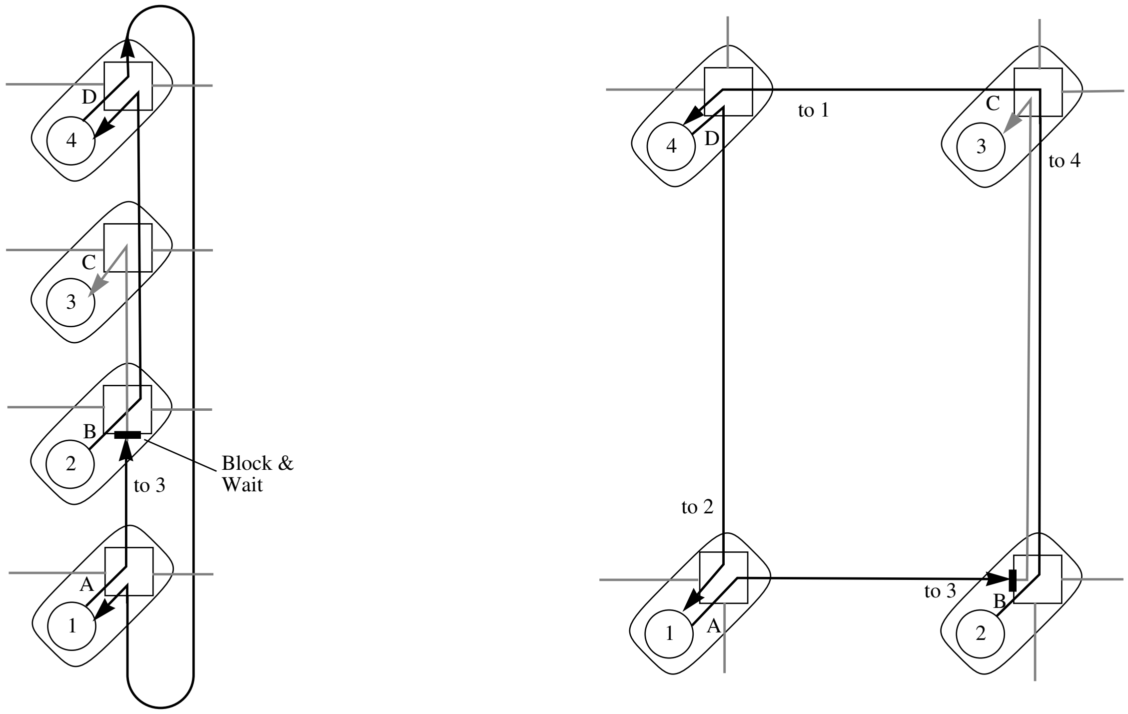

Figure 2: Wormhole routing: Waiting messages in torus loop and a mesh.

3 Sources of routing deadlocks in two dimensional tori

For the designer of a routing algorithm it is essential to know the specific mechanisms that cause deadlocks in the topology he is dealing with. The two dimensional networks used in this report are rectangular tori (if the grid includes wrap around links) and meshes (if the the wrap-around links are not present or not used).

Deadlocks through torus wrap around links

In the network of two dimensional tori there are two different kind of cyclic paths that can lead to deadlocks.

As shown in Figure 3, a single linear loop within the torus can already result in a routing deadlock. This problem can be addressed in two ways.

1. **Surface routing:** Surface routing restricts the routing paths to the surface of the mesh. This is possible on systems with bidirectional links.
2. **Multiple channels:** This method requires at least two channels per link with all buffer resources separately preallocated [DS87].

In the current iWarp system-architecture, an improved version of surface routing is currently preferred over multiple channels. The multiplexed channels and buffer pools provided by the iWarp hardware serve better to decouple general message-passing and specialized systolic communication. The iWarp VLSI component allows both alternatives, since it supports bidirectional links as well as logical channels [Gro89].

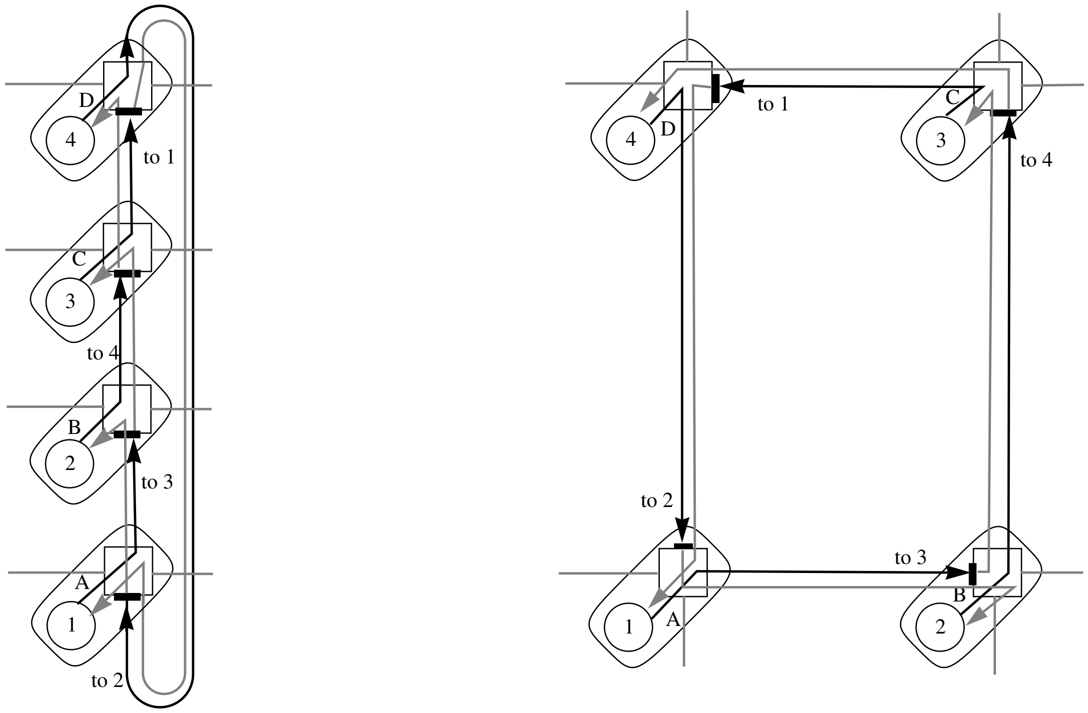

Figure 3: Wormhole routing: Deadlocked messages in torus loop and a mesh.

Deadlocks along circular paths within the surface

Even on routes that do not involve wrap around links of the torus, some simple cyclic paths are possible. Simple cycles can be found within the surface of the mesh. They have either a clockwise and counterclockwise orientation. A deadlock can eventually occur along such a cycle. The surface of most two dimensional torus topologies is restricted to a regular x, y grid due to architectural constraints like backpanel wiring or clocking. Cycle in a rectangular grid are made up by at there are eight characteristic ninety degree corner turns.

Figure 4: The eight corner turns of a grid; four of them clockwise and four of them counterclockwise.

4 Restricting the routing function

The previous chapter outlines how cyclic path are possible and how they can lead to routing deadlocks. Different approaches to prevent deadlocked messages have been studied. One method is to restrict the total number of outstanding messages in the whole array. All token passing schemes follow this approach. In distributed memory systems it is very hard to maintain global constraints on the number of messages while still using all communication bandwidth available.

A better approach for distributed memory systems relies on restricted routes. Messages can travel only a certain way though the network. Most of the previous work on deadlock free message passing has addressed the problem in that way.

4.1 Strongly restricted routers

A routing scheme proposed for hypercubes earlier [DS87] relies on the scheme that all messages traveling on routes following the hypercube-dimensions in decreasing order. Specialized to the simple two dimensional $k \times k$ tori this means that the routing function sends a message first along the Y dimension to the right row and then along the X dimension to the right column. We call that scheme *route Y down, then route X right*.

The simples router for a torus with unidirectional links uses the *route Y down, then route X right*. This method is described and proven to be correct in [DS87]. The main issue of that router is the proper use of wrap around links. A cycle free routing path through every row and column is constructed. A router must be able to generate all path between the cells of that row or column. Since we can move only in one direction (right or down) and it can not be a cycle such a path must be a double loop that visits every cell twice. A realization based on multiple channels per link avoids duplicated wires but still requires every cell to deal with multiple access ports to the loop paths in each dimension.

4.2 Minimally restricted routers

It is easy to see that such a simple *route Y down, then route X right* router is too restrictive to provide all routes within irregular tori, even if it makes use of multiple channels. The example in Figure 5 focuses on a path from an auxiliary cell into the mesh. This path can not be properly routed complying with the restrictions of the unidirectional *route Y down, then route X right* torus scheme.

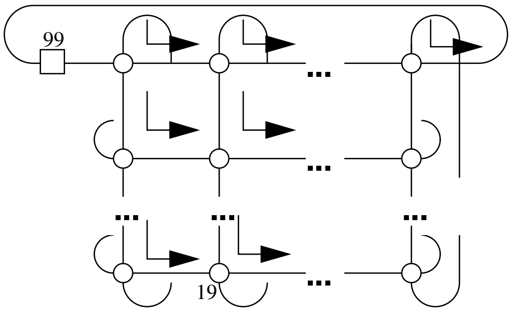

Figure 5: An impossible route (cell 99 to cell 19) in the *route Y down then route X right* scheme.

The problem of the simple router with auxiliary cell lies in the strong restriction to allow messages to turn only from a top-to-down channel into a left-to-right channel. Althought there are eight different types of corner-turns, strongly restrictive routers use only one of them (Figure 6).

Within the regular surface of a mesh, the routing scheme is characterized by the types of corner turns it allows or disallows. There is no need to disallow seven out of eight types of corner-turns for routing within the surface of the mesh. The following theorem states a weaker restriction that is sufficient for deadlock free routing within the regular surface. For the auxiliary cells found in the border different restrictions will be introduced later.

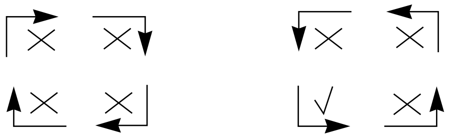

Figure 6: The corner-turns used by the restrictive *route Y then route X* router

Theorem 1 (Forbidden Corner Turns) *To guarantee deadlock free routing within the mesh it is sufficient to disallow one out of four types of corner turns for every orientation, clockwise and counterclockwise.*

Proof 1 *The following argument shows that a deadlocked path necessarily includes all four cornerturns of one orientation. According to the definition of a routing deadlock, the cause of waiting messages can only be busy channels. In a deadlocked message configuration there exists a set of legal routes whose channels form a cycle and are busy. Cycle in planar rectangular grid are either clockwise or counterclockwise and necessarily include all four type of corner turns of that orientation at least once.*

The theorem provides a starting point for the design of less constrained routing methods and for the implementation of the routers needed for practical iWarp systems.

5 Routing functions proven as deadlock free

In many cases a routing problem can be solved by dynamically detecting deadlock situations. Rerouting or resending are used to solve the problem at runtime. This strategy is successfully used by congested railroads, cellular phones or distributed computer systems. Networks of parallel computers deal with large numbers of messages and guarenteed upper bounds on the message latency are very important. Dynamic resolution is therfore not attractive. In static routing methods only a set of restricted routes is considered. This allows to exclude a deadlock by virtue of the router code. The router is verified and proven as deadlock free in the stage of design and coding.

5.1 Proof based on channel dependency graphs

The proof for deadlock free routing on regular hypercubes given in [DS87] is based on channel dependency graphs. The proof constructs the line digraph of the communication network. In a line digraph edges of the original graph become vertices in the digraph, and every pair of edges incident to the same vertex result in an edge of the digraph. The channel dependency graph is contained in the line digraph of the network graph. If a router forwards an incoming message from an incoming channel into an outgoing channel, the incoming channel is said to be dependent on the outgoing channel and therfore has an edge in the channel dependency graph. To show that no deadlock is possible, the channel dependency graph is labeled with numbers.

If the routing functions and the network are simple structures the line-digraph of the network can be drawn easily. The dependency graphs for binary hypercubes remain binary hypercubes.

5.2 A proof based on the labeled network graph

The dependency graph of a practical iWarp torus with auxiliary cells would be large and non-planar. It is almost impossible to manually label the edges of such a graph. It would be highly desirable to have proofs done on the network graph, We consider the network graph to be far more intuitive than the channel dependency graph, and we will develop a direct proof to verify routers on their connectivity graphs with the following definitions and theorems.

A proof of a router is constructed by labeling each channel (an edge in the network graph) with a unique number. Using an order relation on these numbers the routing problem can be restated in the precise language of graphs, order relations and routing functions. A theorem is established to assert that all paths computed by the router follow the labels in an certain order, that there will be no cycles of legal routes and therefore no deadlocks.

Definition 1 (Network Graph) *The network graph (or connectivity graph) of an irregular torus is a directed graph A with a set of nodes N, corresponding to every cell in the network, and a set of edges C, corresponding to each channel.*

$$A := G(N, C)$$

The most important component of a router is the code that determines the route for every possible source and destination. This is the routing function:

Definition 2 (Routing Function) *The routing function RF is a multivalued function that maps a pair of cell addresses <source, destination> to a sequence of channels $c_0 ... c_k$. This sequence of channels denotes the path a message must take from the source cell to the destination cell. Two single valued function can be derived to replace the multivalued routing function. The first channel of a path c_0 is computed by a function R_0 that maps $(N \times N \mapsto C)$. The remaining channel sequence $c_1 ... c_k$ is computed by k successive applications of the channel routing function R. This function R has the the form $(C \times N \mapsto C)$ and maps a previous channel and the destination into the next channel of the path.*

To verify a router all channels of a network are assigned numerical labels.

Definition 3 (Channel labeling) *A channel labeling is a structure of numbers with a non reflexive, total order relation $< (c_i, c_j) := c_i < c_j$. The numbers are assigned one-to-one to the edges of the network graph and cover all edges used by the router. It is often convenient to refer to the complement of the relation as $\geq (c_i, c_j) := c_i \geq c_j$.*

It is not necessary to construct and draw the channel dependency graph as such since we can use the numbered network graph to define the channel dependency relation of a router.

Definition 4 (Channel dependency relation) *A channel dependency relation $\mathcal{R}$ for a given pair of channels is defined as the union of all routing function parts $\mathcal{R}(c_i, c_j) := \{(c_i, c_j) | \exists n \in N : R(c_i, n) = c_j\}$ where N is the set of all possible destinations.*

The channel numberings are carefully chosen according to the structure of the routing function. In our examples the labels are generated as digit strings with a lexicographical order. The relations greater and smaller are obvious and the numbers are easy to write. The following theorem links a labeled network graph to the correctness of a router expressed in terms of cells and channels.

A routing function is deadlock free if all message routes follow the channels along increasing numbers. This is formally stated in Theorem 2.

Theorem 2 *A routing function R is free of deadlock if the channel dependency relation $\mathcal{R}(c_i, c_j)$ is contained in the order relation $< (c_i, c_j)$ established by the numbers of the channel labeling, i.e. $\mathcal{R}(c_i, c_j) \subseteq < (c_j, c_i)$.*

Proof 2 *A routing function constructed from the* $<$ *relation of the labeling routes messages from a current channel only into a channel with a larger number. This is the case because the channel dependency relation derived from the routing function is contained in the relation* $< (c_i, c_j)$ *of the numbering. Since the structure is finite and the order is total, there must be always a channel with a largest number. This channel can not be blocked by any other message since the router can not route a message legally beyond that channel. If there is a message claiming that channel it can be routed into that channel and will eventually be consumed by its destination. Then the channel with the second largest number can not be blocked and a message claiming that channel can be routed. By induction on the labelnumbers every message can be routed and will reach its destination.*

The argument is similar to the one used in [DS87] but is stated in a different way using a labeling for the network graph instead of the channel dependency graph.

Theorem 2 assures that a router is deadlock free but one more property is needed to make it a correct and usable router. To make it deadlock free we restricted the possible paths between cells. We have to make sure that our constraints are not overly restrictive and that there will be at least one path left between each pair of nodes in the network.

Definition 5 (Complete Routing Function) *A routing function is complete if it specifies at least one path between every pair of nodes in the network. With a route defined as a set of chained partial functions, the first part (*$<$*source,destination*$>$ $\mapsto c_0$*) must be defined for every pair of nodes as well as all further parts* $(i = 1..k-1)$ *(*$<c_i$*,destination*$>$ $\mapsto c_{i+1}$*) must be defined.*

In practical arrays some routes between the auxiliary cells might be intentionally omitted, and the completeness requirement is relaxed for some parts of the array (e.g. the routes between different classes of auxiliary cells).

6 Designing a router function

The router function for a given network has to be designed and verified at the same time. The actual code to compute the route is as important as the channel labeling that proves its correctness.

6.1 Coding a router function

A coded router function is a subroutine that accepts source and destination as parameter and returns the route as a machine specific data structure. On iWarp the route is fully determined by the sender, and the data structure returned by the router contains information about initial direction and all corner turns to be taken on that route.

This way of routing is often referred to as *street sign routing* since the data structure describing a route resembles written directions for a city with Manhattan geometry; e.g. "to get from The Village to Avery Fisher Hall: start from Washington Square (on 5th Ave) going uptown to 59th, then turn left and go west to 8th Ave, then turn right, go uptown to 66th street and stop".

The deadlock free router code computes the partial route for each dimension and issues the required corner turns in compliance with the restrictions of the underlying scheme e.g. *route X, then route Y*. The structure of the router program is closely related to the labeling used in the proof of its correctness.

6.2 Verification of a router function

The analysis of corner turns gives us the basic guidelines about possible routing methods and the set of constraints we want to apply. Theorem 1 states the fact that one corner turn per orientation must be disallowed in any labeling.

Picking the disallowed corner turns and constructing a labeling takes care of all routing within the regular part of the network. The regular part of the network of iWarp tori is is the surface of the mesh. The consideration of corner turns leads to general schemes like *route X, then route Y* or *route X left, then Y, then X right*. To deal with warp around links and the irregularities in the border all channels involved in such cases are labeled manually. The labeling allows specific routes into or around the irregularity. The label assignment is done in the network graph and therfore is easy to do.

Inside the regular part of the network, consistency of the router code with the labeling is shown by structural induction. Outside the regular part, manual verification of routing function might be necessary. It is important that the constraints are not chosen to be too strong. They must leave at least one path between every pair of cells (completeness requirement).

Both completeness and compliance with the numbering scheme can be verified exhaustively at design time with a simple test program calling the prototype of the routing code repeatedly. Such a test program has to verify whether all routes exist and check them against the channel labels. In a two dimensional array with n nodes there are n^2 routes with an average length $\frac{\sqrt{n}}{2}$ so the verification of a router takes $O(n^2 \times \sqrt{n})$ comparison steps. Note that the number of possible deadlock configurations is exponential in the size of the network n, since every subset of messages has to be considered. Further the sequential order in which these messages are sent off must be taken into account as well. An analysis of all message configurations possible in network is likely to be intractable. This underlines the need for constraints on routing functions or number of messages.

7 Routing methods for practical arrays

7.1 A basic *route X then route Y* router

The limitations of the basic deadlock free router for hypercubes were described earlier. This basic router will be extended to the irregular case and we therefore restate the algorithm and its properties here.

function router(s,d:cellid) route:
route X on shortest path (left or right), then route Y on shortest path (up or down)

7.1.1 Properties of the *route X then route Y* router

- Works on the regular surface of the network.
- Can not include any auxiliary cells.
- Exhibits congestion in the center of the mesh.

7.1.2 The labeling of the *route X then route Y* router

The channels are assigned numbers between 2.0b.0a and 2.1d.1c; the front most digit 2 is common to all channels since all are in the surface of the regular mesh. The numbering is essentially same as in Figure 10, but without auxiliary cells and wrap around links.

Channels in	carry labels:
x+ direction (left)	2.0y.0a to 2.0y.0c
x- direction (right)	2.0y.1a to 2.0y.1c
y+ direction (down)	2.1x.0b to 2.1x.0d
x- direction (right)	2.1x.1b to 2.1x.1d

Additional rules
The identifiers x,y specify the column or the row location of the channel in the network. They are not critical for routing but assure that conveniently that the channel numbers are unique withing the whole network; 1x is used for vertical channels and 0y for horizontal channels.

The x+ channels are numbered left to right using numbers from a to c, the x- channels are numbered right to left using numbers from c to a. The same numbering applies for y-,y+ numbers.

Theorem 3 *The route X then route Y shortest path router always routes messages from channels with lower numbers into channels with higher numbers.*

Proof 3 *The router first injects a message from a cell into the network. The router deals then with the x-part of the destination address sending the message left or right. By definition of the numbering above the message will encounter only increasing channel numbers as long it is traveling straight on the shortest path towards the destination or the point of a corner turn. If the message arrives at the corner turn the router will now deal with the y-part of the destination address in the same way. Since all y-channels have a higher number than the x-channels the corner turn from X to Y-direction results in an increase of channel numbers. Once the message is at its destination the router delivers the message to the computation agent.*

Just using the definitions of the router algorithm and the numberings, we have shown that messages travel always along channels with increasing numbers, and by Theorem 2 the router is therefore deadlock free. In the new routing schemes derived in the following paragraphs we will limit ourselves to show the existence of the numbering in a figure rather to formally state it.

7.2 One turn routing

Some configurations of iWarp have only one auxiliary cell, the host interface cell. As stated in the introduction the multiple channels are not available to the router and unless stated otherwise the functions are restricted to surface routing. The routing function designed for this case is simple and elegant. Given the source and destination addresses in x_{src}, y_{src} and x_{dest}, y_{dest} the route is computed with the following rule:

If $y_{dest} > y_{src}$
then route X, turn and then route Y
else route Y, turn and then route X

7.2.1 Properties of the *one turn* router

- Messages between any two cells travel the same way forth and back.
- No message must turn at corners more than once.

7.2.2 The labeling of the *one turn* router

The channel labeling was constructed in two steps. $0a.0y$ to $0c.0y$ labels are assigned to all horizontal channels from left to right, $0x.1b$ to $0x.1d$ to all vertical channels from bottom to the top, $1a.1y$ to $1c.1y$ labels cover all channels from right to left, and finally the numbers $1x.0b$ to $1x.0d$ are assigned to the channels from the top to bottom. The row identifiers $0x$ and $1x$ of the vertical channels are chosen to lie lexicographically in between the labels of the horizontal channels left and right of them. The labeling for an example 4×4 torus is shown in Figure 8.

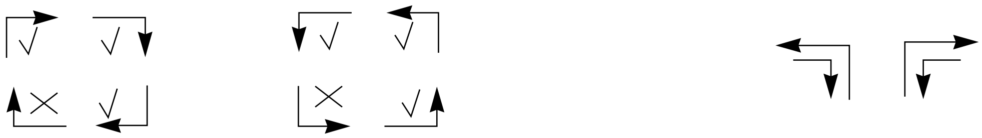

Figure 7: The labeling used to proof the *one turn* router allows more corner turns [left] than actually required by the routing function [right].

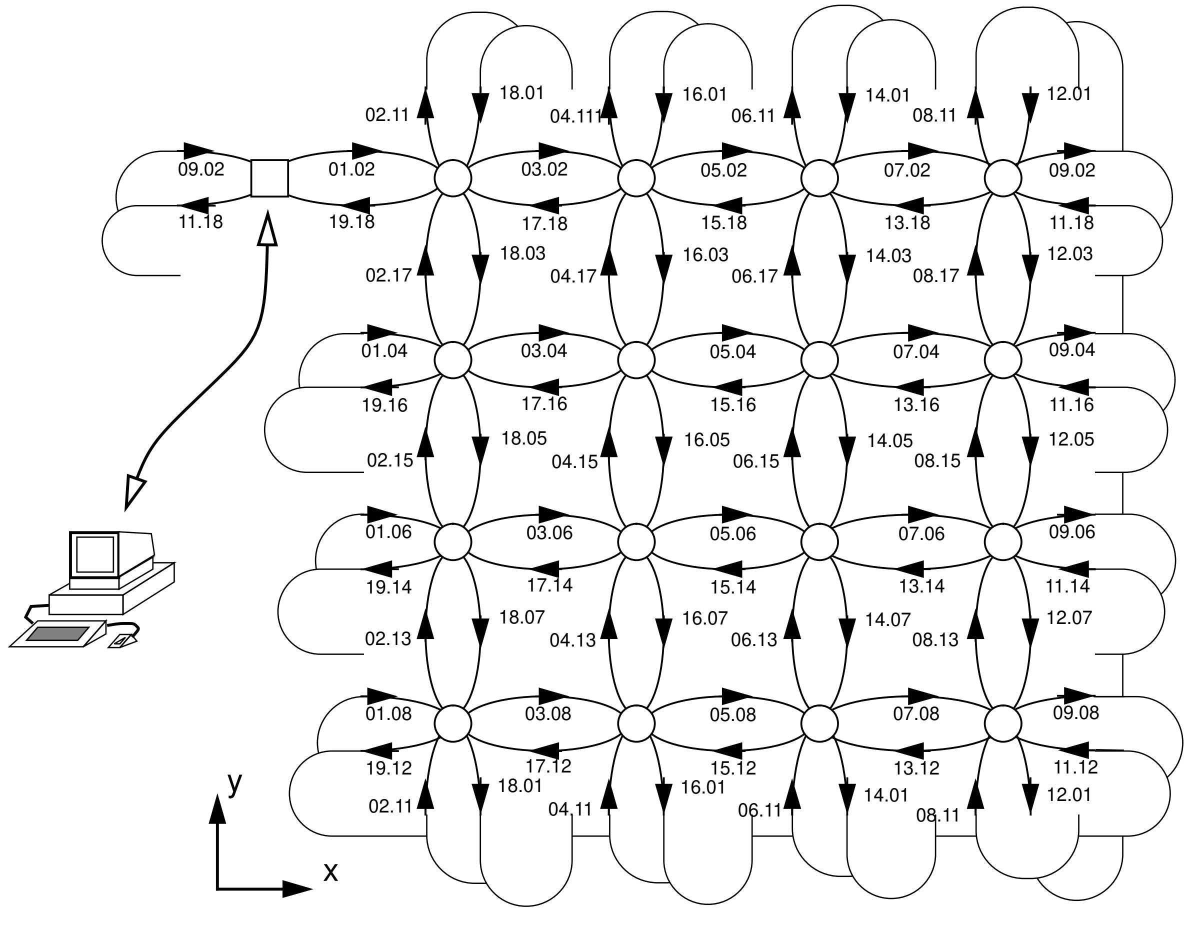

Figure 8: The channel labeling for the *one turn* routing scheme.

Theorem 2 and the labeling in Figure 8 prove that deadlock is impossible. To apply Theorem 2 all paths computed by the router must follow the channels with increasing labels. The proof is a straight forward program verification of the router code with the structure of the labeling.

7.3 Routing with many auxiliary cells along the borders.

Sometimes more than one auxiliary cell must be placed into a two-dimensional torus. It is best to place all of them into two arbitrary bisections of the torus; one bisection horizontally and one vertically. The two bisections divide the array into the border and the regular surface of the mesh (see definition in Figure 1.

Within the regular surface of the mesh the proposed method routes strictly according to the rule *route X then route Y*. The connections *from* auxiliary cells to the mesh are seen as an artificial dimension W regardless whether they are physically in X or Y direction. Analogous, all connections from the mesh *to* the auxiliary cells are labeled as Z dimension. The modified scheme is described as:

Route W, if applicable then route X followed by Y finally route Z if applicable

7.3.1 Properties of the routes

- Messages travel the dimensions strictly in one order.
- Within the surface message will turn at most once at corners, from and to auxiliary cells at most twice.

7.3.2 The labeling

A channel numbering can be systematically constructed from the description of the routing method. The most significant digit reflects directly the dimension. 1 for the W dimension, 2 for the X and Y dimensions, 3 for the Z dimension. The priority of X over Y is expressed in the second most significant digit of the lexicographical label. Figure 9 shows a 4×4 array as an example of that labeling.

7.4 Routing with cells inserted into the surface of the mesh

Routing functions will stay simple as long as all partially connected cells are aligned to the border (i.e. placed into wrap around links of the torus). If there are not enough places available in the border of the mesh, some auxiliary cells must be inserted into the surface of the mesh. It is still possible to construct routing functions for these cases.

7.4.1 Properties of the routes

- To achieve a complete routing function, all inserted auxiliary cells have to be fully connected over all four channels, two of them in the X- and two of them in the Y- dimension.

7.4.2 The labeling

The labelings of both previously described routers can be modified to deal with inserted cells in certain cases. The method describe here applies if every inserted auxiliary cell creates a new row and column. Figure 10 shows the router of Section 7.2 modified for one inserted cells.

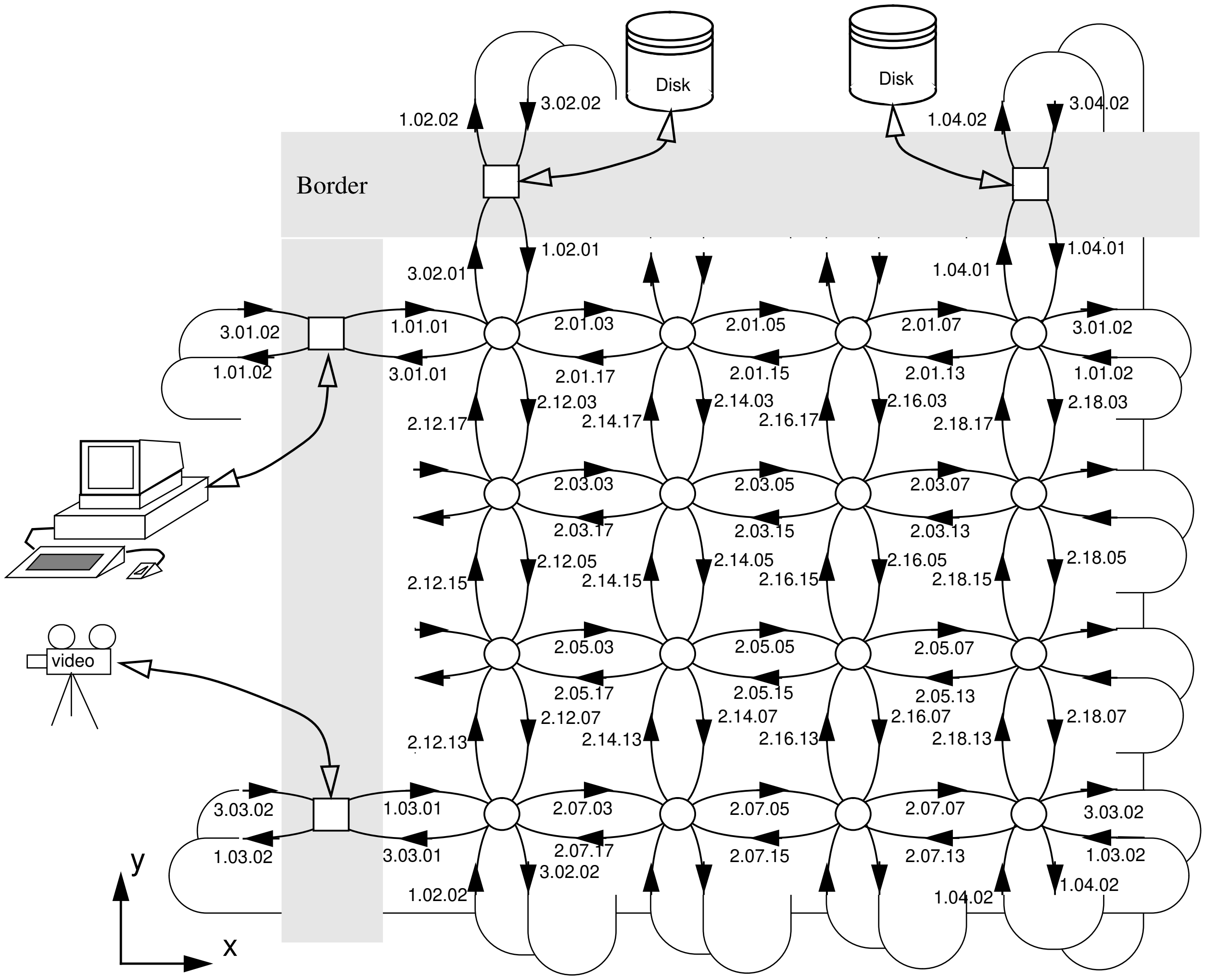

Figure 9: The channel numbering to proves the basic routing scheme for the case of *auxiliary cells within the border*

Cells left and below the insertion point are renumbered. At the grid points where an inserted column or row has no cell, the two horizontal or vertical arcs are replaced by a single arc which may carry either of the labels. (e.g. channel 05.02 and channel 07.02 of Figure 7 form a new single channel labeled 07.02 since the node at the former grid location (2.0) has been dropped to accommodate an auxiliary cell).

8 Routing schemes with better balanced load distributions

In the previous section the flexible method to design and verify routing functions has been applied to irregular meshes. The flexibility of less constrained routes and unused wrap-around links can be used in other ways as well. In heavily loaded networks routing messages along all paths in the surface of the mesh will cause hot spots. Better routing functions try to balance the load distribution more evenly over the network.

Our improved routing schemes are able to route all I/O message traffic over the less heavily loaded links along the borders. The term I/O message traffic will be use to denote all messages from and to the auxiliary

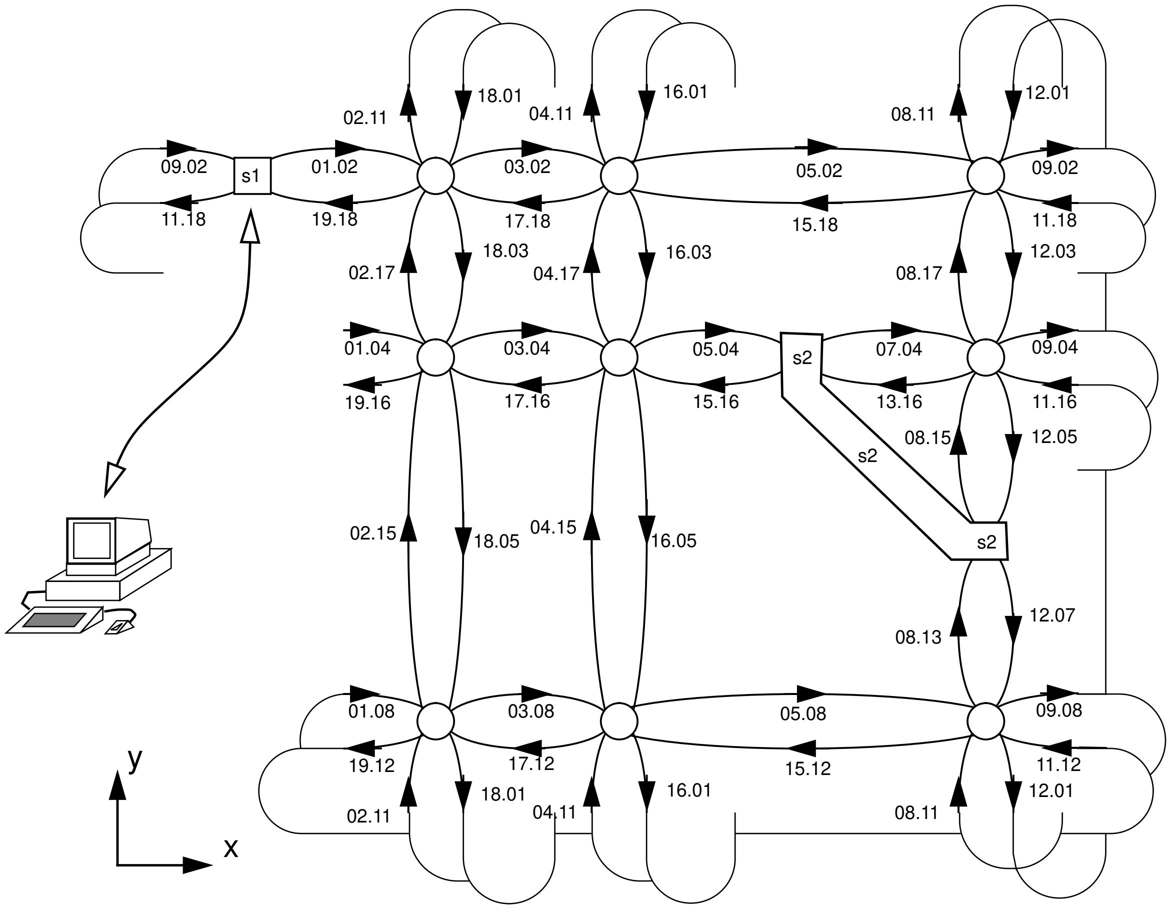

Figure 10: The channel numbering which proves the *inserted cells* routing scheme.

cells. To improve the load balance in the routers of our iWarp networks, some previously unused wrap around links are labeled appropriately in the W and Z dimensions. A routing scheme with a better balance of I/O traffic is obtained by applying the following routing rule:

case route is:
within mesh-
  route X followed by Y
from auxiliary cell to mesh-
  route over W channels to the corner closest to destination
  route X then Y to destination
from mesh to auxiliary cell-
  route X then Y to the closest corner
  route over Z channels to destination

The channel labels are assigned so that from the auxiliary cell each of the four corners can be reached over W dimension links. The auxiliary cell can be reached from every corner over Z dimension links. An example is shown in Figure 11.

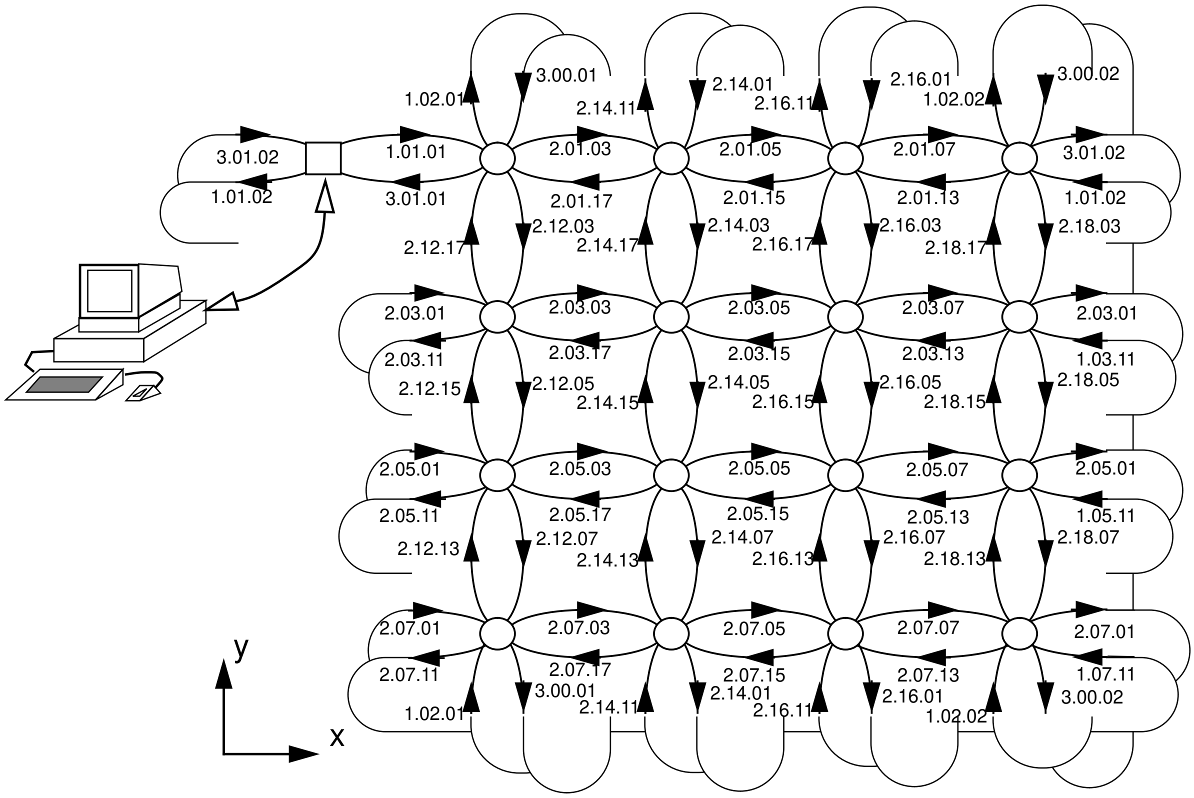

Figure 11: The channel numbering for routing with *better balanced I/O traffic*.

8.1 Analysis of the message traffic

Different routers show different distributions of the message load over the channels involved. For an analysis and evaluation the message traffic is counted and averaged over a certain set of messages. It will be assumed that every cell tries to send a certain number of data elements to every other cell. The algorithm sends n^2 messages between every pair of cells. This message pattern could either result from a naive n^2 sorting algorithm or more likely from a probabilistic algorithms that averages in that case.

The number of messages traveling across every link is counted and percentages are computed for the routers in Section 7.1 and 7.2 using an example of a 4×4 torus. Figures 12 considers only traffic among the computation cells but shows that the different routing functions described in Section 7.2 and 7.3 will lead to different message loads (congestions). The characteristic load distribution of I/O message traffic by itself is shown separately in Figure 13.

Better methods of load balancing could be derived from routing schemes. Most labelings provide a limited freedom for adaptive routes, and therefore multiple shortest routes between two nodes could be taken, even in a deadlock free router. In Figure 8 any possible shortest stair route from upper right to lower left might be chosen. Multiple channels could be used to create less constraint and more adaptive routers. For this study we limit ourselves to a single channel but attempt to balance the traffic of I/O messages and the traffic among computation cells. Figure 14 shows that the message distribution for random traffic within the mesh [Figure 12] and for traffic relating to I/O [Figure 13] complement each other well under the condition that the total I/O traffic is roughly one quarter of the computation traffic. The summed percentages are renormalized and drawn in Figure 13.

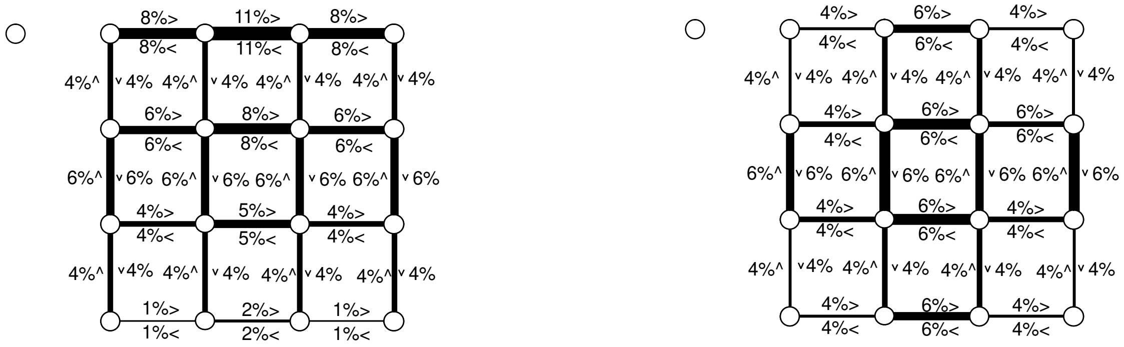

Figure 12: Message traffic load distributions within the two different routing schemes. *One turn routing* [left] and *route Y then route X* [right].

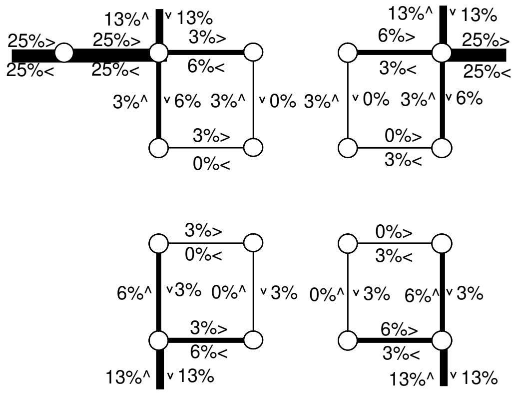

Figure 13: Load distributions for I/O message traffic only.

A typical computation involves both kind of traffic and can be balanced reasonably well.

8.2 Balanced traffic for multiple I/O cells

The method for I/O traffic balancing generalizes to configurations with multiple auxiliary cells. To obtain complete routing functions all auxiliary cells have to be connected among themselves, Figure 15 gives an example.

Additional links create a subnetwork linking the auxiliary cells. Theses links are considered to be additional dimensions (e.g. U, Z1) and labeled with 0.xx and 4.xx numbers. Messages from an auxiliary cell travel through that subnetwork to the corner closest to the destination cell in the surface. Correspondingly, messages to the auxiliary cells leave the surface through the closest corner and travel over the subnetwork to the auxiliary cell addressed.

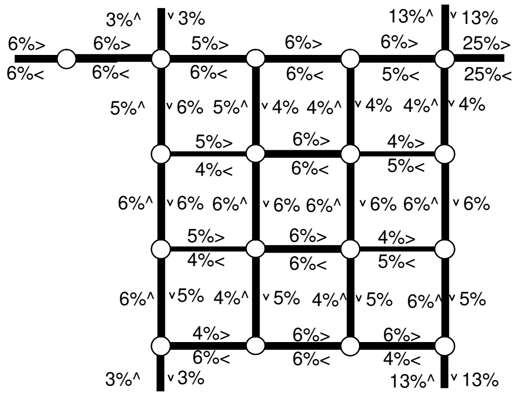

Figure 14: Well balanced load distribution for a surface router and a well designed I/O message router.

9 Conclusion

We have shown that the standard methods for deadlock free routing on hypercubes fail to provide certain routes to auxiliary cells and are therfore not applicable to irregular tori. The previously known methods have been used as a starting point to construct new routers. A proof working on the network graphs instead of channel dependency graphs simplified the design and the verification of routing functions significantly for irregular topologies. For practical iWarp configurations (two dimensional tori with attached cells), several deadlock free routing functions with different properties were designed and discussed in detail. These new routers are able to deal with irregularities and some aspects of load balancing. For the routing module of the current iWarp system software we proposed and implemented a simple and elegant solution that uses at most one corner turn per route. For system expansions we proposed router functions dealing with multiple auxiliary cells attached to the surface of the mesh and one for auxiliary cells inserted into the surface of the mesh. The less constrained routing functions were also successfully used to statically balance the load of messages between I/O messages traffic and communication within the surface.

10 Acknowlegements

I would like to thank my faculty advisor at Carnegie Mellon, Professor Thomas Gross, for his suggestions, assistance and encouragement to write up this report. Many thanks also to the iWarp System Software group at Intel Corp., Beaverton, Oregon for their suggestion to work on the iWarp routers. I am also grateful to all my fellow students for the comments they gave me, after listening to my seminar talks or reading the drafts of my manuscript.

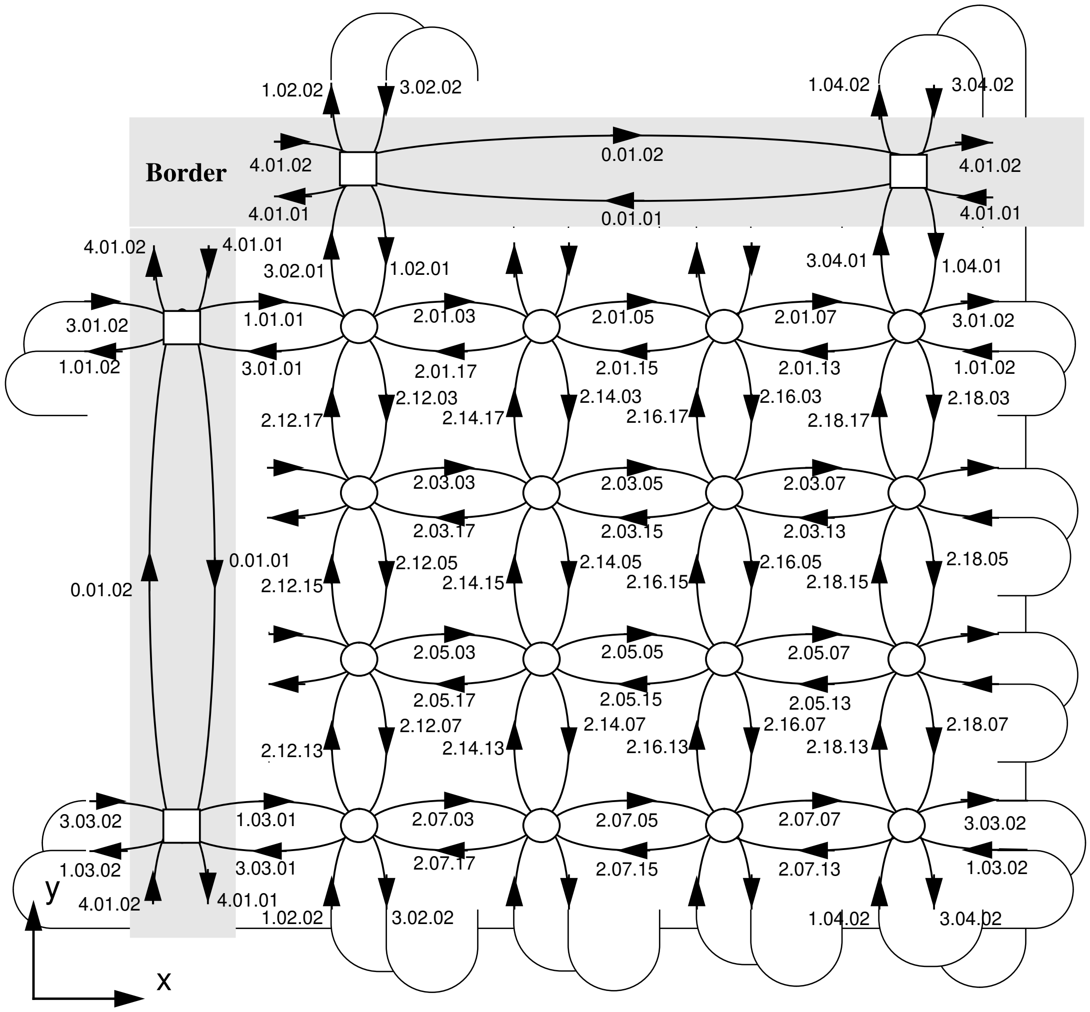

Figure 15: The channel labeling which proves the *load balancing with multiple I/O cells*

References

[DS87] W. Dally and C. Seitz. Deadlock free message routing. *IEEE Transactions on Computers*, Vol C-36, May 1987.

[Gro89] Thomas Gross. Communication in iwarp systems. In *in Proceedings of Supercomputing '89*, pages 436–445, Reno, NV, USA, November 1989. IEEE Computer Society and ACM SIGARCH.

[HMS+86] J.P. Hayes, T.N. Mudge, Q.F. Stout, S. Colley, and J. Palmer. The architecture of the hypercube. In *in Proceedings of the 1986 International Conference on Parallel Processing*, pages 653–660, August 1986.

[iWarp88] S. Borkar, R. Cohn, G. Cox, S. Gleason, T. Gross, H.T. Kung, M. Lam, B. Moore, C. Peterson, J. Pieper, L. Rankin, P.S. Tseng, J. Sutton, J. Urbanski, and J. Webb. iwarp: An integrated solution to high-speed parallel computing. In *Proceedings of Supercomputing '88*, pages 330–339, Orlando, FL, USA, November 1988. IEEE Computer Society and ACM SIGARCH.

[iWarp90] S. Borkar, R. Cohn, G. Cox, T. Gross, H.T. Kung, M. Lam, B. Moore, W. Moore, C. Peterson, J. Susman, J. Sutton, J. Urbanski, and J. Webb. Supporting systolic and memory communication in iwarp. In *Proceedings of 17th Intl' Symposium on Computer Architecture*, May 1990.