Evaluation of Load Sharing in HARTS while Considering Message Routing and Broadcasting (Extended Abstract)

Kang G. Shin　　Chao-Ju Hou
Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122

1 Introduction

In this paper, we apply the load sharing (LS) mechanism proposed in [1, 2] to HARTS, an experimental distributed real-time system [3] currently being built at the Real-Time Computing Laboratory of the University of Michigan.

The nodes in HARTS are interconnected by a C-wrapped hexagonal mesh [4] and coordinated under the LS mechanism to evenly share "overflow" tasks. The HARTS routing and broadcasting schemes in [4, 5] are used for transferring tasks and broadcasting state-changes. The virtual cut-through switching scheme implemented in HARTS [6] is used for inter-node communication. The interconnection network and all the schemes used for task routing, state-change broadcasting, and message-passing are considered as an integrated part of the LS mechanism. An overview of HARTS is given in Section 2. Components of the LS mechanism, i.e., *buddy sets*, *preferred lists*, and *region-change broadcasts*, are introduced in Section 3.

By exploiting/integrating features of the above schemes for/into LS, we rigorously analyze the performance of LS in HARTS while considering all LS-related communication activities. In particular, we analytically evaluate the integrated LS performance using the probability of dynamic failure, P_{dyn} — the probability of a node failing to complete a task before its deadline — as a yardstick. The analysis methodology is outlined in Section 4. The paper concludes with Section 5.

2 Overview of HARTS

HARTS is an experimental distributed real-time system, where a set of Application Processors (APs) along with a Network Processor (NP) form a node. These nodes are interconnected via a C-wrapped hexagonal mesh topology [4]. The APs execute computational tasks, and the NP (which contains a custom-designed router, buffer memory, a RISC processor, and the interface to APs) handles both intra- and inter- node communications.

The work reported in this paper was supported in part by the National Science Foundation under Grant MIP-9203895, and the Office of Naval Research under Grants N00014-92-J-1080.

C-wrapped H-meshes have several nice properties as reported in [4]. First, C-type wrapping results in a simple, transparent addressing scheme, where the center node is labeled as node 0, and the other nodes are labeled in sequence along the horizontal direction. (An example of the addressing for an H-mesh of dimension 5 is shown in Fig. 1.) Second, C-type wrapping results in a homogeneous network. Every node may view the mesh as a set of concentric hexagons with itself at the center. Consequently, all nodes are topologically equivalent, and the wrapped mesh becomes homogeneous. Third, the diameter of an H-mesh of dimension e, denoted as H_e, is $e-1$. Consequently, any routing/broadcast packet traverses at most $e-2$ intermediate nodes before reaching its destination node. Fourth, simple and efficient routing algorithm, broadcast algorithm, and switching scheme can be devised, as discussed in [4, 5, 6]. We exploit all features of the routing and broadcasting algorithms, and the virtual cut-through switching scheme (which support LS-related communication activities) in our analysis.

3 Load Sharing Mechanism

We use the LS approach in [1, 2] as the distributed scheduling mechanism in HARTS, and describe the associated LS transfer, location, and information policies.

Transfer Policy: upon arrival of a task at node i, the node checks whether or not the cumulative execution time (CET) is greater than the task laxity[1] or not. If it is (i.e., node i is capable of completing the task in time), the task will be accepted and queued for execution. Otherwise, the task will be transferred (in the form of routing packets) to a receiver node chosen by the location policy.

Location Policy:Preferred Lists and Buddy Sets To reduce the possibility of more than one node simultaneously transferring their overflow tasks to the same destination node, we exploit the topological properties and the homogeneity feature of a C-wrapped H-mesh, and order all the other nodes according to the distance from node i into the *preferred list* of node i, $\forall i$, under the restriction that every node in an H_e is selected as the kth preferred node of one and only one other node, for $k = 1, ..., 3e(e-1)$.

Once each node's preferred list is constructed, the node's *buddy set* can be formed by taking the first N_B nodes from

[1]The laxity of a task is defined as the latest time a task must start execution in order to meet its deadline.

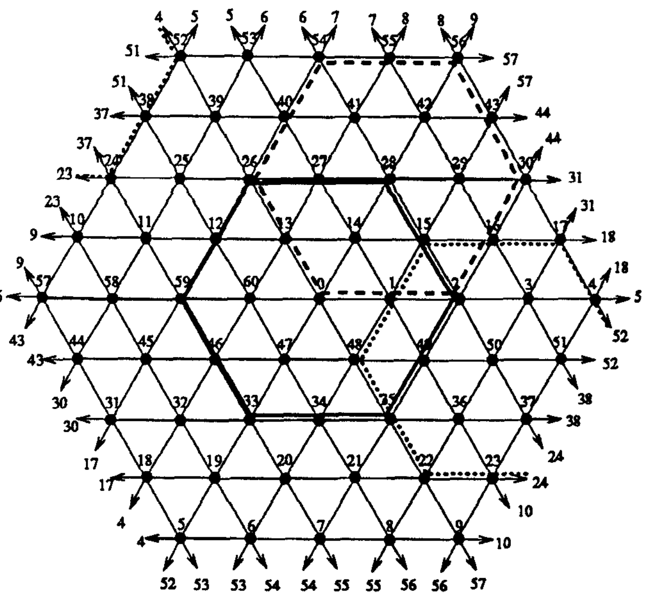

Figure 1: A C-wrapped hexagonal mesh of dimension 5, H_5. Also shown is how the buddy sets of node 0, node 28, and node 50 overlap with one another.

the top of its preferred list. A node with an overflow task can then select the *first* node found in its buddy set that is capable of completing the task in time, and transfers its overflow task to that node.

By using the buddy sets constructed above, one can reduce the traffic overhead, because each node i is restricted to communicate with only a set of N_B nodes in its proximity. Moreover, these buddy sets overlap with one another (Fig. 1) so that an overflow task may be transferred from an overloaded node to some other node which is a member of a different buddy set.

Information Policy: Region-Change Broadcasts
Each node exchanges state information via region-change broadcasts. Specifically, K_T state regions defined by (K_T-1) thresholds, TH_1, TH_2,..., TH_{K_T-1}, are used to characterize the workload of each node. Each node broadcasts a message, informing all the other nodes in its buddy set of its state-region change whenever its state crosses TH_i for some $i \in \{1,\ldots,K_T-1\}$. The state information kept at each node is thus up-to-date as long as the broadcast delay is not significant.

4 Performance Analysis

We first construct an embedded continuous-time Markov chain to describe task arrival/transfer/completion activities under the proposed LS mechanism in HARTS. With the homogeneity property of the wrapped H_e, the general methodology — which was proposed and verified (via simulations) in [7] for homogeneous systems — of first modeling the state evolution of a single node in isolation and then combining node-level models into a system-level model (by including task activities at the system level in the parameters of node-level model) is used.

Two sets of parameters are derived from the constructed Markov model: (1) the probability density function of QL, $p_N(n)$, $n \geq 0$, and (2) the rate of transferring tasks, λ_{TT}, the rate of state-change broadcasts, λ_{SC}, and the probability of transferring an overflow task to a node h hops away, q_h. The latter set of parameters is fed into the queueing network, where each nodes forms a $G/M/1$ queue, that models the handling of (both task-transfer and broadcasting) packets at each node in HARTS and characterizes the hexagonal mesh topology and the virtual cut-through switching scheme implemented in HARTS.

The probability density function of packet delivery time, $f_{D_t}(t)$, can then be derived from the queueing network model, which, along with $p_N(n)$, $n \geq 0$, is used to derive the probability density function of task waiting time (i.e., the time a task is queued for execution plus the delay the task experiences if it is transferred), $f_{W_k}(t)$. Finally, the probability of dynamic failure, P_{dyn}, can be computed from $f_{W_k}(t)$.

5 Concluding Remarks

The analysis presented in this paper is essential to the design of any LS mechanism for real-time applications. First, the model gives a quantitative measure of traffic overheads (through λ_{TT}, λ_B, and $f_{D_t}(t)$) introduced by the LS mechanism. Second, one can study the effects of varying LS design parameters (e.g., the size of buddy sets, N_B, and the threshold values, TH_i, for state-change broadcasts) on the performance and thus fine tune the parameter values to minimize P_{dyn}. (This problem is currently under investigation.)

The analysis methodology presented in the paper can be extended to other LS mechanisms and other interconnection topologies. To extend this methodology to other LS mechanisms (topologies), one only needs to properly derive the parameters which characterize the transfer policy and the location policy. Once they are determined, the derivation of the others follows the same approach.

References

[1] K. G. Shin and Y.-C. Chang, "Load sharing in distributed real-time systems with state change broadcasts," *IEEE Trans. on Computers*, vol. C-38, pp. 1124–1142, Aug. 1989.

[2] K. G. Shin and C.-J. Hou, "Design and evaluation of effective load sharing in distributed real-time systems." To appear in *IEEE Trans. on Parallel and Distributed Systems*, 1993.

[3] K. G. Shin, "HARTS: A distributed real-time architecture," *IEEE Computer*, vol. 24, pp. 25–34, May 1991.

[4] M.-S. Chen, K. G. Shin, and D. D. Kandlur, "Addressing, routing, and broadcasting in hexagonal mesh multiprocessors," *IEEE Trans. on Computers*, vol. 39, pp. 10–19, Jan. 1990.

[5] D. D. Kandlur and K. G. Shin, "Reliable broadcast algorithms for HARTS," *ACM Trans. on Computer Systems*, vol. 9, pp. 374–398, Nov. 1991.

[6] J. W. Dolter, P. Ramanathan, and K. G. Shin, "Performance analysis of message passing in HARTS: a hexagonal mesh multiprocessor," *IEEE Trans. on Computers*, vol. C-40, pp. 669–680, June 1991.

[7] D. L. Eager, E. D. Lazowska. and J. Zahorjan, "Adaptive load sharing in homogeneous distributed systems," *IEEE Trans. on Software Engineering*, vol. SE-12, no. 5, pp. 662–675, 1986.