Ragasa: A High-bandwidth, Low-latency and Reliable In-network Broadcast Algorithm for Dragonfly Network

Junchao Ma, Dezun Dong, Zihao Wei and Zhen Ruan

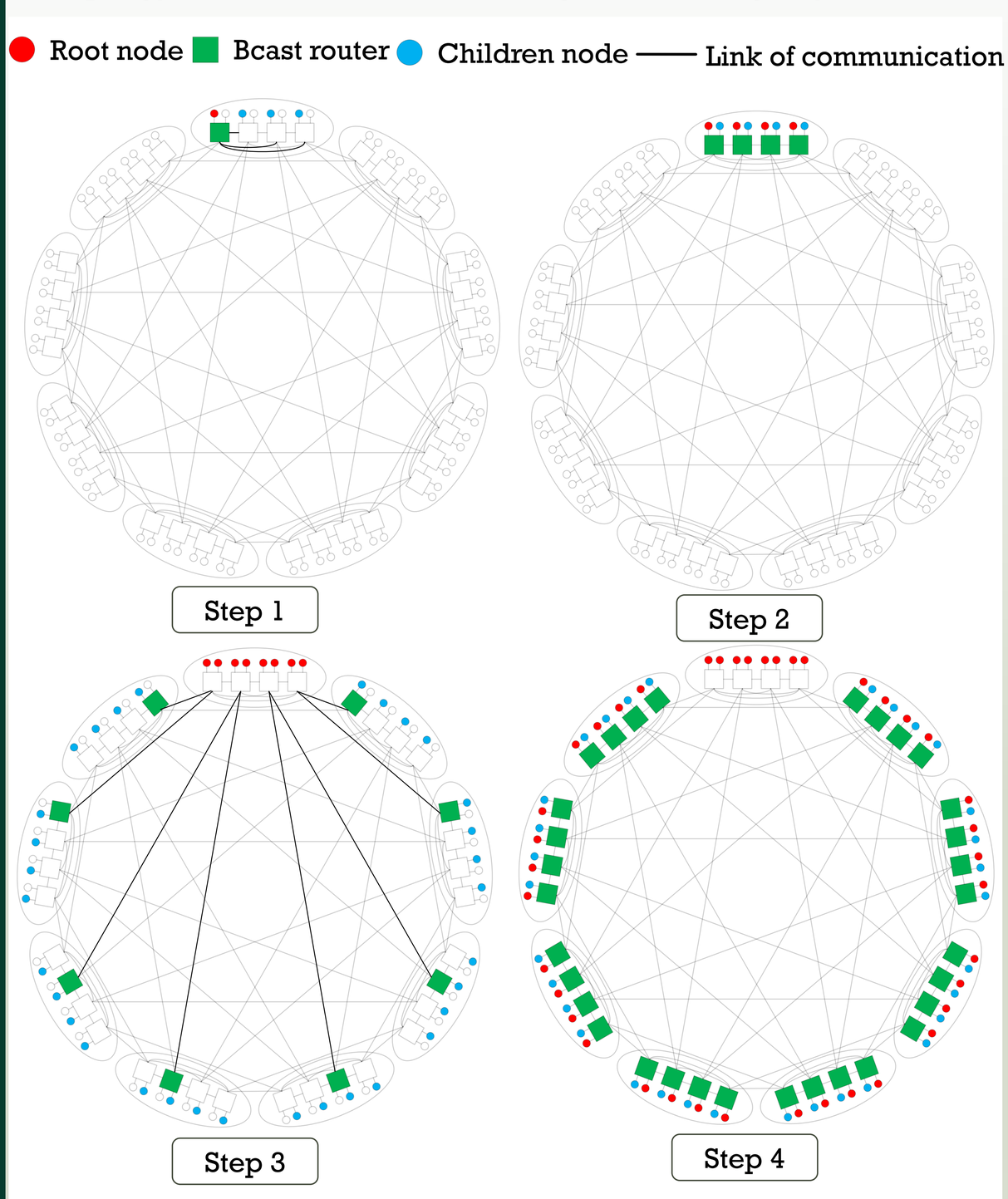
Laboratory of High-Performance Networking and Architecture, National University of Defense Technology, China

Introduction

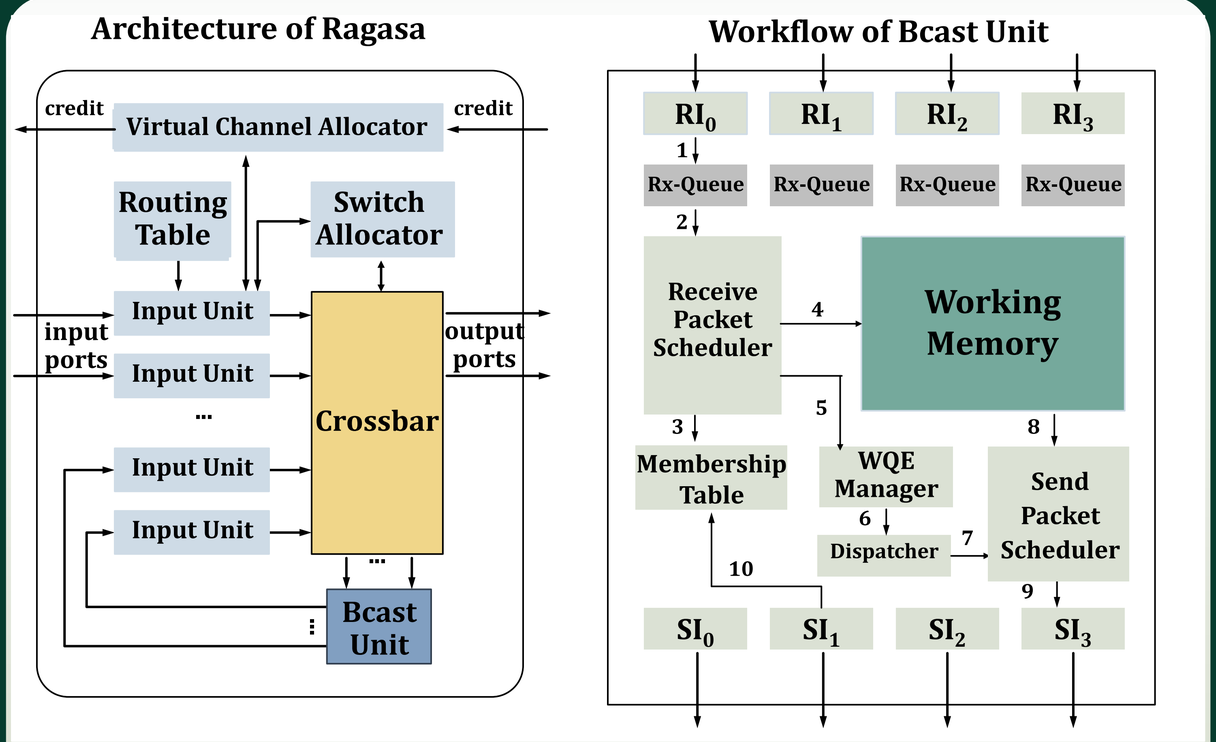
MPI Bcast is a fundamental collective operation crucial to many high-performance computing applications. Various broadcast algorithms have been implemented in the message-passing interface (MPI) standard, such as OpenMPI , MPICH , and MVAPICH. Recently, the network offload acceleration solution has emerged as a desirable and practical approach to accelerate computation and communication concurrently by enabling network computation during data transport. For example, SHArP protocol developed by Mellanox[1] leverages InfiniBand hardware multicast to send a single message from a host to multiple end nodes by replicating it in the switches when necessary. This paper proposes Ragasa, an adaptively tuned and dragonfly topology-aware design for broadcast based on router's offload, which significantly reducing redundant network traffic.

Methods

Topology-aware In-network Broadcast Algorithm for Dragonfly Network

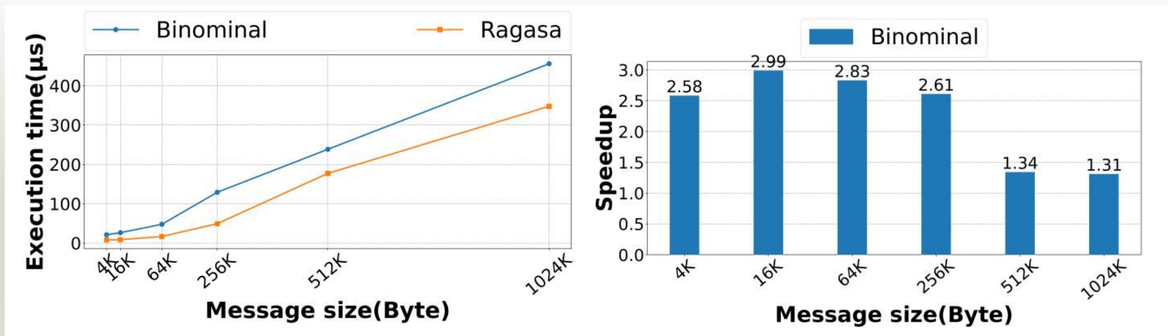


Contention-free broadcast tree : The broadcast of a communication group is decomposed into multiple smaller broadcasts. We establish a priority order for broadcast tree grouping and matching: 1) nodes reside in different groups; 2) nodes are within the same group; and 3) nodes are under the same router. Starting from the root node, data is first broadcast via routers with broadcast offloading capability in the source group to nodes matching priority 1). Subsequently, the nodes that received the data in the first step utilize routers with broadcast offloading capability within their respective groups to broadcast the data to other nodes in the same group, according to priority 2). Finally, the child nodes of the broadcast tree that received the data in the second step disseminate the data to other directly connected nodes via the broadcast functionality of their directly connected routers.



The management node sends a Mini Packet (MP) to each router participating in broadcast offloading. This MP carries the child node information of the destination router. A router that receives the MP stores this information in its Membership Table. When flits arrive at the Receive Interface (RI), they are assembled into packets and then forwarded to the Rx-Queues for storage (Step 1). The Receive Packet Scheduler selects a packet from a non-empty Rx-Queue (Step 2). If it is an MP control packet, the scheduler parses the packet data, generates membership information from the carried data, and stores it in the Membership Table. If it is a data packet, the scheduler verifies whether there is sufficient Working Memory to store the data. If sufficient space is available, the data is stored in the Working Memory (Step 4). If insufficient space is available, the data packet is not dequeued from the Rx-Queue and awaits rescheduling. Once a sufficient amount of broadcast data is received, descriptors are generated based on the member information in the Membership Table and sent to the Work Queue Element to await processing (Step 5). The descriptors are then queued in the Dispatcher for processing (Step 6). Successfully established connection descriptors are sent to the Send Packet Scheduler (Step 7). The Send Packet Scheduler retrieves the corresponding data from the Working Memory based on the information in the descriptors, assembles it into packets, and forwards them to the Send Interface (SI) (Step 9). The SI segments the packets into flits and injects them into the network. After data transmission is complete, a completion signal is sent to the Membership Table, and the corresponding memory is released (Step 10).

Results & Discussions



Df(2,4,2); Execution time of different algorithms on a 72-node dragonfly network.

Df(2,4,2); Ragasa's speedup compared with the Binominal algorithm on a 72-node.

Conclusions

When all node perform Broadcast with middle message sizes (e.g. 16KB), Ragasa achieves a 2.99X speedup over Binominal. As the message size increases, Ragasa performs similarly to Binominal for large messages. We will try to experiment more when message size is small in the future.

References

1. Richard L et al. 2016. Scalable Hierarchical Aggregation Protocol (SHArP): A Hardware Architecture for Efficient Data Reduction. *In 2016 First International Workshop on Communication Optimizations in HPC (COMHPC). 1–10.* DOI:10.1109/COMHPC.2016.006

Acknowledgements: The authors would like to gratefully acknowledge the support from the Laboratory of High-Performance Networking and Architecture (*HiNA*) at National University of Defense Technology. This work is supported by the National Key Research and Development Program of China under GranNo,2022YFB4501702,and the National Natural Science Foundation of China under Grant No.62202482 and No,U24B20151.

Junchao Ma,
PhD Student, NUDT
Tel: +8618711102844
Email: majunchao@nudt.edu.cn