Research and Design of a Deadlock-Free Large-Port Multicast Acceleration Engine: Postprint

Authors: Wang Dawei, Hu Nongda

Date: 2016-11-02T00:00:00+00:00

Abstract

The rapid growth of network scale poses greater challenges to interconnect networks in four aspects: scalability, high performance, reliability, and low-power design. High-radix switch chips, while improving network scalability, can reduce communication latency, network power consumption, and cost, becoming one of the development directions for system interconnect networks. The expansion of network scale also leads to increased collective communication overhead. This paper primarily investigates design strategies for efficient multicast acceleration engines for high-radix switch chips. It presents the necessary and sufficient conditions for deadlock-free multicast in multi-stage interconnect networks, and accordingly proposes a multicast acceleration engine architecture employing a central arbiter with separated data and control. Combined with synchronous and asynchronous multicast methods, this architecture can achieve very high saturation throughput across a wide fanout range.

Full Text

Preamble

High-Radix Deadlock-Free Multicast Acceleration Engine Research and Design
Dawei Wang, Nongda Hu

Abstract

The rapid growth of network scale presents greater challenges for interconnection networks in four key aspects: scalability, high performance, reliability, and low-power design. High-radix switch chips can effectively improve network scalability while reducing communication latency, network power consumption, and cost, making them a key direction for system interconnection networks. Expanding network scale also increases collective communication overhead. This paper

investigates design strategies for efficient multicast acceleration engines tailored for high-radix switch chips. We present necessary and sufficient conditions for deadlock-free multicast in multistage interconnection networks and propose a multicast acceleration engine architecture employing a centralized arbitrator with separated data and control paths. By combining synchronous and asynchronous multicast methods, the design achieves high saturation throughput across a wide fanout range.

Keywords: multistage interconnection network; high-radix switch; multicast; deadlock

1 Introduction

The insatiable demand for computational power in high-performance applications drives parallel computer systems to continuously improve single-processor performance while expanding system scale. The Dawning 5000A petascale computer has already reached 1,650 nodes, and future multi-petascale systems will approach or exceed tens of thousands of nodes. Designing high-performance, highly scalable ultra-large-scale interconnection networks has become critical to the success of future cluster systems.

Dramatic system scale growth challenges interconnection networks across four dimensions: scalability, high performance, reliability, and low-power design. High-radix switch chips can effectively enhance network scalability, reduce network diameter, decrease the number of switch chips and network links, and thereby lower network latency, power consumption, and cost while improving reliability [1][2][3]. Consequently, they have become a key technology for building ultra-large-scale cluster interconnection networks. In scientific computing—the primary application layer for ultra-large-scale parallel systems—there exists substantial demand for collective communication operations, including multicast, broadcast, all-to-all, barrier synchronization, and all-reduce [4][5]. Among these, multicast is the most representative pattern. Multicast communication involves sending identical messages from a single source node to multiple destination nodes. This pattern is widely adopted in scientific computing and serves as a critical sub-step for other collective operations such as barrier synchronization [6][7] and all-reduce [7]. Therefore, high-radix multicast engines represent one of the key technologies for high-radix switch chips.

The primary challenge in high-radix switch chip design is meeting scalability requirements with limited resources. Integrating both high-radix unicast crossbars and multicast acceleration engines within the same chip demands that each functional module be concise and efficient. Deadlock phenomena occur in multicast communication. This paper proposes a resource request waiting graph (RWG) methodology for analyzing multicast deadlocks and derives necessary and sufficient conditions for deadlock-free multicast in multistage interconnection networks. Based on these conditions, we present a centralized arbitrator employing a data-control separation strategy that ensures deadlock-free multi-

cast while efficiently implementing multicast forwarding with minimal resources.

2 The Multicast Deadlock Problem

Deadlock fundamentally stems from resource limitations—when requesters hold resources while requesting new ones, mutual waiting occurs among requesters for resources held by others, preventing forward progress. All packets within a deadlock configuration become permanently blocked in the network, unable to reach their destinations, while newly arriving packets also become blocked by deadlocked packets, eventually paralyzing portions or the entire network. Therefore, deadlock must be avoided or recoverable.

In multistage interconnection networks (MINs), unicast channel dependency graphs are inherently acyclic, making unicast communication deadlock-free [8]. However, multicast introduces resource dependencies between different ports, creating multicast deadlock problems in MINs. Two deadlock scenarios exist: intra-switch deadlock and inter-switch deadlock at the same stage.

[Figure 1: see original paper] illustrates these multicast deadlock configurations. Figure 1(a) depicts intra-switch multicast deadlock: when input buffer IB0 requests output buffers OB0 and OB1 but only obtains arbitration permission for OB0, while input buffer IB1 requests OB0 and OB1 but only receives permission for OB1, both IB0 and IB1 request resources held by the other while refusing to release their own held resources, resulting in indefinite waiting.

Figure 1(b) illustrates inter-switch deadlock at the same stage: since channel L12 already occupies channel L14 and L22 occupies L24, while L12' s multicast output channel L13 from the same group requests L24 (already held by L22), and L22's multicast output channel L23 requests L14 (already held by L12), deadlock occurs if L12 and L13, as well as L22 and L23, must output simultaneously and cannot operate independently.

2.1 Resource Request Waiting Graph

To formally analyze multicast deadlock, we propose the resource request waiting graph methodology. An RWG is a directed graph $G = (V, E)$, where V represents network resources such as buffers and channels (buffer resources for intra-switch analysis and channel resources for inter-switch analysis), and E represents request-wait relationships between nodes. E contains three edge types: arbitration edges, request edges, and multicast port edges. A directed edge (i, j) is an arbitration edge if resource i has obtained resource j through arbitration. It is a request edge if resource i is requesting resource j. Multicast port edges depend on communication system characteristics: if the system requires ports from the same multicast to output together and cannot output independently (synchronous multicast, also called non-splitting), the RWG contains multicast port edges—meaning if (i, j) are output ports of the same multicast, directed edges (i, j) and (j, i) are multicast port edges. These edges belong to E. If the

system allows independent output of different ports from the same multicast (asynchronous multicast, also called splitting), no multicast port edges exist.

Thus, in synchronous multicast RWGs: - If resource i has obtained resource j: $(i, j) \in E$ - If resource i is requesting resource j: $(i, j) \in E$ - If i and j are output ports of the same multicast window: $(i, j), (j, i) \in E$

In asynchronous multicast RWGs: - If resource i has obtained resource j: $(i, j) \in E$ - If resource i is requesting resource j: $(i, j) \in E$

Based on this RWG definition, the resource request waiting graphs for the two deadlock scenarios correspond to [Figure 2: see original paper]. With synchronous multicast, Figure 2(a) contains a resource cycle $IB0 \rightarrow OB0 \rightarrow IB1 \rightarrow OB1 \rightarrow IB0$, preventing packet forwarding and creating deadlock. If the network employs synchronous multicast, Figure 2(b) contains a cycle $L13 \rightarrow L12 \rightarrow L14 \rightarrow L23 \rightarrow L22 \rightarrow L24 \rightarrow L13$, causing system deadlock. With asynchronous multicast, L12 and L13, as well as L23 and L22, can multicast independently–L12 can transmit its multicast packet, and upon completion, L23 can acquire channel L14 to continue sending another packet. Similarly, L13 can continue transmission, eliminating resource interlocking. Correspondingly, the asynchronous RWG lacks edges $(L12, L13)$ and $(L22, L23)$, and thus contains no cycles.

Lemma 1: If a cycle appears in the resource request waiting graph of an interconnection network at any moment, deadlock occurs in the network at that moment.

Proof: Assume the network contains a cycle. Without loss of generality, let resource P_1 in the cycle hold resource P_2 while requesting resource P_0, with P_2 reachable to P_0 in finite steps, meaning P_2 possesses an occupancy relationship to P_0. Thus, the resource requested by P_1 is held by a resource it already occupies. No node in the cycle will automatically release held resources, causing perpetual cyclic waiting and resulting in deadlock.

The resource request waiting graph expresses dependency relationships within a switch chip or across an interconnection network at a given moment, varying according to different packet requests and arbitration conditions.

2.2 System Model Description

Before proving the necessary and sufficient conditions for deadlock-free multicast in virtual-cut-through MINs, we establish fundamental system attributes as preconditions:

Attribute 1 (Packet Format Transmission): Any node can send packets with maximum length MTU (Maximum Transmission Unit) to any number of destination nodes.

Attribute 2 (Packet Transmission Atomicity): Any communication resource on the data path, such as buffers or physical link channels, once allo-

cated to a packet header, will continuously transmit the entire packet until completion.

Attribute 3 (Virtual-Cut-Through Switching): The system employs virtual-cut-through switching, requiring each switch node' s input buffer to accommodate at least one complete MTU-sized packet.

Attribute 4 (Arbitration Fairness): Arbitration policies in switch chips are fair without starvation phenomena.

Attribute 5 (Packet Termination): Packets are processed within finite time upon reaching their destinations.

2.3 Necessary and Sufficient Conditions for Deadlock-Free Multicast in Virtual-Cut-Through MINs

This section presents necessary and sufficient conditions for deadlock-free multicast in virtual-cut-through multistage interconnection networks.

Lemma 2: In a multistage interconnection network using virtual-cut-through switching, if multicast deadlock occurs, the packet headers causing deadlock must be located in and only in the same switch chip.

Proof: Let packets P_i and P_j have headers S_i and S_j respectively, located in different stage switch chips, with stage(S_i) > stage(S_j). Examining P_i' s transmission at stage(S_j), according to Attributes 1, 2, and 3, if P_i' s header resources have been allocated a data path, the remaining portions will pass through the stage(S_j) switch within at most MTU/BW time (where BW is port bandwidth) and reach stage(S_i) of the MIN. Therefore, packet P_i cannot deadlock with P_j at stage(S_j).

Further considering the possibility of deadlock between P_i and other packets (without loss of generality, P_k) during transmission at stage(S_i): if P_k' s header is not at stage(S_i), the situation resembles the initial relationship between P_i and P_j. According to Attribute 5, packets entering their destinations will eventually be consumed and links will become idle. Thus, P_i cannot deadlock at stage(S_i) with packets whose headers are not in the same stage switch chip (e.g., P_k).

Consequently, throughout the transmission process, a packet cannot deadlock with packets whose headers are not in the same stage switch chip—meaning deadlocked packets must have their headers in the same stage switch chip.

Theorem 1: In a multistage interconnection network using virtual-cut-through switching, the necessary and sufficient condition for deadlock-free multicast is that no cycles exist in the resource request waiting graph within any switch chip at any moment.

Proof: According to Lemma 2, deadlocks in virtual-cut-through MINs can only occur within switch chips. Therefore, we need only prove that within a switch

chip, the sufficient condition for deadlock-free multicast is the absence of cycles in the resource request waiting graph at any moment.

Sufficiency: By the contrapositive theorem of propositional logic ($A \rightarrow B \Leftrightarrow \neg B \rightarrow \neg A$), to prove the proposition "If no cycles exist in any switch chip' s resource request waiting graph at any moment, then the virtual-cut-through MIN has no multicast deadlock," we can prove its equivalent: "If the virtual-cut-through MIN has multicast deadlock, then cycles exist in the switch chip' s resource request waiting graph."

Assume deadlock occurs within a switch chip at some moment. By definition, requesters must occupy some output ports while requesting others. Without loss of generality, let the switch have n output ports represented by set $\{0, 1, ..., n-1\}$, and m resource requesters represented by set $\{a_0, a_1, ..., a_{m-1}\}$. In deadlock, $m \leq n$.

[Figure 3: see original paper] illustrates the relationship between requester a_i, granted port set G_i, and request set R_i. For any requester a_i, there exist two sets: granted port set G_i and waiting request set R_i, where $G_i \subseteq \{0, 1, ..., n-1\}$ and $R_i \subseteq \{0, 1, ..., n-1\}$.

By the deadlock definition—where some requesters holding resources request new ones, causing mutual waiting and preventing forward progress—let x be the number of requesters satisfying $G_i \cap R_j \neq \emptyset$ for $i \neq j$. By definition, $x > 0$. According to the RWG definition, we have $a_i \rightarrow G_i$ and $R_i \rightarrow a_i$. Since there are n ports and m requesters, $x < mn$.

Consider two cases: - When $0 < x < m$: There are $x + 1$ elements on both sides of a_i. If no duplicate elements exist in these $x+1$ elements, all packets can complete transmission within finite time, contradicting the deadlock assumption. Therefore, duplicate elements must exist, meaning the RWG contains a cycle. - When $m \leq x \leq mn$: The mapping from R_i to a_k has at least m possibilities, while there are $m + 1$ total elements on both sides of a_i. By the pigeonhole principle, placing $m + 1$ elements into m values necessarily yields duplicate elements, meaning cycles exist in the RWG.

Thus, the sufficient condition holds.

Necessity: By contrapositive, to prove "If the virtual-cut-through MIN has no multicast deadlock, then no cycles exist in any switch chip' s resource request waiting graph," we prove the equivalent: "If a cycle exists in a switch chip' s resource request waiting graph, then multicast deadlock occurs in the virtual-cut-through MIN." This is precisely Lemma 1.

Therefore, Theorem 1 holds.

2.4 Deadlock Avoidance Methods

Theorem 1 shows that if the resource request waiting graph within the same switch chip is acyclic, the entire MIN is deadlock-free. To prevent deadlock,

we employ centralized arbitration: within each switch chip, a unique central arbitrator checks resource dependencies to ensure the RWG remains acyclic.

For synchronous multicast, the central arbitrator verifies whether each requester can simultaneously obtain all requested resources. If possible, it grants all requested resources at once; otherwise, it grants none. This prevents cycles in the RWG, as shown in [Figure 4: see original paper].

For asynchronous multicast, the central arbitrator' s function can be relaxed. Since each requester can transmit upon obtaining partial resources regardless of whether it continues requesting other resources, the data transmission no longer depends on obtaining all resources. This logically equates to not requesting remaining resources. Therefore, the central arbitrator simply allocates resources to requesters as available, ensuring deadlock-free multicast.

3.1 Hardware Multicast Architecture Research

Multicast crossbars differ from unicast crossbars primarily in their ability to replicate packets. Packet replication can be implemented in non-splitting or splitting modes. Splitting offers better performance but more complex control logic. Most multicast microarchitecture research extends unicast architectures, focusing on various scheduling policies.

3.1.1 Knockout-Based Multicast

The Knockout architecture employs output queuing and supports both unicast and multicast communication [9]. Its basic principle broadcasts multicast packets via a bus, with each packet header containing an identifier (ID) for the output destination port group. Each destination port uses a fast filter to check the ID and forward the packet if matched, discarding it otherwise. The Knockout structure has two main limitations: (1) The Knockout algorithm assumes the probability of more than L packets simultaneously arriving at a port is extremely low, enabling N-to-L concentrators. However, this assumption often fails under heavy multicast loads, causing significant packet loss. (2) Output queuing suffers from poor buffer scalability, requiring $(L+1) \times B$ bandwidth. For high-radix switches, L increases with N, making this impractical for high-performance switch designs.

3.1.2 Input-Queued Multicast Scheduling

Since input queuing is hardware-implementable, existing work focuses on input-queued structures to develop multicast scheduling algorithms that improve saturation throughput while reducing switching latency.

Reference [10] proposes scheduling policies TATRA and WBA based on a single FIFO input queue structure. While efficient, these have two shortcomings: (1) High scheduler design complexity makes hardware implementation difficult;

(2) Suboptimal multicast saturation throughput performance, with uniform distribution results never exceeding 90%. Reference [11] presents the FIFOMS scheduling algorithm based on Virtual Output Queuing (VOQ). Although it achieves 100% saturation throughput under uniform distribution with fairness guarantees, VOQ requires $2N$ buffer queues for high-radix switches—an impractical structure.

3.1.3 Buffered Crossbar Queuing Multicast Scheduling

Reference [12] studies the MXRR multicast scheduling algorithm for single-input-queue buffered crossbar structures. This method focuses on output port scheduling, achieving high saturation throughput for uniform distributions but poor performance under bursty traffic. Reference [13] proposes a buffered crossbar structure with multiple input queues, examining the impact of input queue count, crossbar capacity, and various scheduling methods on multicast saturation throughput. While buffered crossbar structures achieve good performance, they require at least $2N$ buffer queues, making them infeasible for high-radix implementations.

3.2 Central Arbitration Multicast Engine Design

Our multicast engine design employs a single-input-queue structure with separated data and control flows. A unified central arbitrator provides arbitration for each requester, preventing deadlock while significantly saving arbitrator resources and effectively improving throughput.

We first propose a basic high-radix multicast crossbar design using 64 ports as an example, then present improved designs in three aspects, establishing gate-level cycle-accurate simulation models for each to analyze performance variations. This section focuses on the microarchitecture design, with performance evaluation in the following section.

3.2.1 Basic Central Arbitration Multicast Engine

The basic central arbitration multicast engine comprises four main functional modules: input buffers, packet schedulers, central arbitrator, and multiplexers. The architecture is shown in [Figure 5: see original paper]. Input buffers cache multicast packets using FIFOs. The packet scheduler: (1) reads packet headers to determine multicast request port lists; (2) sends request signals to the central arbitrator with the requested port list; (3) upon receiving grant, sends packets to the multiplexer; otherwise, continues waiting for authorization.

The central arbitrator: (1) maintains priority queues based on fairness principles; (2) evaluates whether packet scheduler requests meet conditions and grants qualified requests; (3) configures multiplexers to establish data paths for successful arbitrations; (4) maintains Resource Available Registers (RAR). The 64-bit RAR indicates output port status (1 = available, 0 = unavailable). When

the central arbitrator allocates an output port to a scheduler, the corresponding RAR bit is cleared; when the multiplexer notifies completion, the bit is set. The central arbitrator grants requests when the scheduler has highest priority and all requested ports are available. Upon granting, it allocates all requested ports simultaneously to ensure deadlock freedom. The multiplexer: (1) selects data according to central arbitrator configuration; (2) notifies the central arbitrator upon completion to update RAR.

While this basic design effectively prevents multicast deadlock, it has limitations: (1) FIFO input buffers cause head-of-line blocking—if the head packet lacks arbitration permission, it blocks all subsequent packets even when their requested output ports are idle; (2) Allocating all resources at once inevitably wastes port capacity and arbitration cycles; (3) With many ports, the central arbitrator' s single-cycle evaluation of one scheduler simplifies logic but wastes arbitration cycles and opportunities. Researching methods to evaluate multiple schedulers per cycle could improve throughput. The following sections propose solutions to these issues.

3.2.2 Multi-Queue Optimization Strategy

To reduce head-of-line blocking probability, we add a packet dispatcher and multiple waiting queues to the basic structure, modifying the packet scheduler while keeping other modules unchanged. The adjusted structure is shown in [Figure 6: see original paper].

The packet dispatcher: (1) reads packets from input buffers and obtains output request port lists from headers; (2) checks waiting buffer availability and whether existing packets' output port lists conflict with the current request. To preserve packet ordering, packets cannot be moved to waiting buffers if conflicts exist. If space is available and no conflict exists, the dispatcher moves the packet from the input buffer to a waiting queue. Waiting queues use FIFO structures and must provide their stored packets' output request ports to both dispatcher and scheduler. With waiting queues, the scheduler changes: (1) after a waiting queue fails to obtain central buffer arbitration for certain cycles, the scheduler switches to another valid waiting queue based on priority; (2) to prevent deadlock, each active waiting queue has a timer—when overflow occurs, the queue continues waiting without rotation even if arbitration is not obtained within a certain period.

Key research questions for multi-queue structures: (1) How many waiting queues are needed for good performance? (2) How should queue replacement timing be set, and what is its relationship with packet length?

3.2.3 Split Transmission Optimization Strategy

Multi-queue structures mitigate head-of-line blocking but still waste output ports because the central arbitrator uses an all-or-nothing allocation to prevent

deadlock. Conflict probability increases significantly with larger fanout coefficients, reducing overall output throughput. To improve multicast throughput, we allow requesters to transmit multicast packets in batches. If partial resources are obtained, transmission continues while requests for other resources are withdrawn; otherwise, the central arbitrator might reallocate output ports, causing logical errors.

[Figure 7: see original paper] shows the split-transmission multicast engine microarchitecture, which adds transmit buffers and modifies the packet scheduler and central arbitrator. Transmit buffers require write-once-read-multiple functionality, necessitating RAM-like storage instead of FIFOs. Buffer clearing awaits commands from the packet scheduler.

In this structure, the packet scheduler: (1) reads transmit buffer packet headers to obtain request output port lists and requests arbitration; (2) initiates transmission on granted ports, calculates remaining incomplete ports, and withdraws current arbitration requests; (3) after current transmission completes, requests remaining ports from the central arbitrator; (4) when all output ports of a multicast packet complete transmission, sends a command to clear the current packet and prepares for the next.

The central arbitrator changes to inform the packet scheduler which requested output ports are granted when arbitrating. For split transmission mode, we experimentally investigate whether this approach effectively improves multicast saturation throughput in low-to-medium fanout ranges.

3.2.4 Multi-Arbitration Optimization Strategy

Considering hardware feasibility, the central arbitrator can only evaluate one requester per clock cycle. For 64 ports, this serialization impacts performance, particularly for small packet forwarding where arbitration waiting cycles constitute a relatively high proportion of total transmission time, resulting in lower throughput. To further improve multicast throughput while maintaining hardware feasibility, we can increase the number of arbitrations per cycle. Compared to the basic design, multi-port arbitration primarily modifies the central arbitrator to evaluate multiple packet schedulers per cycle, leaving other components unchanged.

The key research question is determining the appropriate number of arbitration requests to process per cycle while balancing throughput performance and hardware implementability.

3.2.5 Hybrid Optimization Strategy

The aforementioned improvement strategies target different performance limitations of the basic model and are not mutually exclusive. Combining them can better improve multicast switching throughput and reduce latency. Therefore, studying performance parameters under hybrid strategies is necessary. Key hy-

brid approaches include: (1) multi-queue structure combined with split transmission (multi-transmit-buffer split transmission); (2) all three strategies combined (multi-transmit-buffer, multi-arbitration split transmission).

4 Performance Evaluation

To analyze multicast switching engine performance, we established gate-level cycle-accurate simulation models for each structure. This section presents the model and performance evaluations of various optimization strategies.

4.1 Simulation Model

The multicast switching engine uses a 64-port design. Each port injects packets at a constant rate, with throughput varying from low load to switch saturation. Multicast packet fanout (number of output ports) ranges from 2 to all 64 ports. Packet headers contain destination port lists, with destination selection following a uniform distribution. Simulation consists of warm-up and steady-state phases. Steady-state is achieved when input traffic multiplied by fanout approximately equals output traffic, remaining stable for a substantial duration. Statistics are sampled only after reaching steady-state.

4.2 Basic Central Arbitration Multicast Engine Performance

We evaluate saturation throughput and average latency across different fanout coefficients and packet lengths for the basic engine, as shown in [Figure 8: see original paper].

Saturation throughput exhibits a U-shaped trend across three stages. Stage 1 (fanout 8): throughput decreases linearly with fanout. Stage 2 (fanout 9-23): throughput fluctuates within a small range (<5 percentage points). Stage 3 (fanout 24-64): throughput reaches its minimum at fanout 24, then gradually increases beyond 32 ports. Maximum saturation throughput increases with packet length due to lower overhead ratios. For 1KB packets, maximum saturation throughput reaches 97.71%; for 1.5KB packets, 98.46%.

Average latency under saturation shows two distinct phases. For fanout < 24, latency grows approximately linearly with fanout. Beyond fanout 32, latency reaches maximum and remains constant. At small fanouts, similar saturation throughput combined with increasing fanout causes latency to grow multiplicatively. Above fanout 32, only one multicast packet can output per cycle while others wait, making saturation throughput independent of fanout.

The basic central arbitration strategy' s strong relationship checking to prevent deadlock prevents ideal throughput across wide fanout ranges. Head-of-line blocking also limits performance improvements at small fanouts, necessitating alternative strategies.

4.3 Multi-Queue Structure Performance

Multi-queue structures increase arbitration success probability by providing more comparable packets while maintaining strong deadlock checking. Using 512B packets, [Figure 9: see original paper] shows throughput and latency curves for different numbers of transmit buffers.

Multi-queue structures primarily improve saturation throughput for fanout 24 (stages 1-2). Beyond fanout 24, the probability of sending only one multicast packet approaches 1, rendering multi-queue structures ineffective. For fanout < 24, they effectively increase comparable multicast packets, improving throughput. However, throughput gains diminish with more waiting buffers. Two buffers improve throughput by up to 10 percentage points over one buffer; three buffers add 4.4 percentage points over two; three to four buffers achieve performance close to 16 buffers (difference <4 percentage points). This occurs because while more buffers increase the probability of finding conflict-free packets, they also reduce idle ports, creating an upper bound on performance gains. Two to four waiting buffers offer the best cost-effectiveness. Multi-queue structures have minimal impact on average latency under saturation.

Another key parameter is queue switching cycles—the waiting period before attempting arbitration for the next queue when a multicast packet conflicts with currently transmitting packets. Using 512B packets and three waiting buffers, [Figure 10: see original paper] shows performance across switching cycles. Smaller switching cycles yield higher throughput, with cycle=1 performing best and cycle=32 worst, though the impact is limited (maximum improvement <8 percentage points). A 3-queue structure with 1-cycle switching delay improves saturation throughput by up to 15 percentage points for packet lengths 64B-1.5KB at fanout < 24, with minimal latency impact.

Since multi-queue structures still employ strong relationship checking, saturation throughput remains suboptimal at large fanouts, motivating research into breaking this constraint.

4.4 Split Transmission Structure Performance

Split transmission allows multicast packets to be sent in batches. The central arbitrator can partially grant requested ports based on availability, further improving throughput. [Figure 12: see original paper] shows multicast throughput and latency for split transmission mode.

Saturation throughput increases gradually with fanout and packet length. For 512B packets, throughput exceeds 85% at fanout 16 and reaches over 90% at fanout 32. This improvement occurs because conflicting multicast packets can transmit in multiple batches without waiting for all output ports to be available, enabling multiple packets to transmit simultaneously. Larger fanout covers more output ports, increasing throughput.

[Figure 13: see original paper] compares multi-queue and split transmission

structures for 512B packets. At fanout=2, split transmission performs worse (13 percentage points lower) due to wasted arbitration cycles and reduced efficiency from partial transmissions. Multi-queue structures excel at small fanouts with low conflict probability and full-port forwarding per successful arbitration. However, as fanout increases, split transmission' s advantage grows, improving throughput by 7-50 percentage points for fanout 4-56 by breaking the all-or-nothing constraint. Since split transmission throughput stabilizes above fanout 16, average latency grows linearly with fanout.

4.5 Multi-Arbitration Structure Performance

Improving arbitration efficiency—evaluating multiple requesters per cycle—can increase throughput. However, hardware complexity limits per-cycle arbitration count. We investigate suitable values balancing throughput and implementability.

[Figure 14: see original paper] shows saturation throughput for packet lengths 64B-1.5KB with 1-6 arbitrations per cycle (many curves overlap). Multi-arbitration only benefits 64-128B packets at fanout 4. For 64B packets, 2 arbitrations/cycle improves throughput by 20 percentage points over 1 arbitration, while 3 arbitrations adds only 1.3 percentage points over 2. For 128B packets, the improvement is 4.6 percentage points (2 vs. 1) and 0.3 percentage points (3 vs. 2). For 256B packets, multi-arbitration provides negligible benefit.

At large fanouts, output port contention probability is extremely low, making multiple arbitrations meaningless. At small fanouts, arbitration overhead constitutes a significant portion of small packet transmission time, so improved arbitration efficiency helps. For large packets, arbitration serialization impact is minimal. Multi-arbitration is only effective for small packets, with 2 arbitrations/cycle offering the best cost-effectiveness.

4.6 Hybrid Acceleration Performance

The optimization strategies are complementary and can be combined. We evaluate hybrid approaches, focusing on multi-queue plus split transmission (multi-transmit-buffer split transmission) and the full combination of all three strategies.

For the multi-transmit-buffer split transmission structure, [Figure 15: see original paper] shows 512B packet simulation results. With 2 transmit buffers, small-fanout throughput improves significantly (>11 percentage points); 3 buffers add 3.8 percentage points over 2; 4 buffers add only 1.6 percentage points over 3. Performance gains diminish with more buffers, with 6 buffers achieving <1 percentage point difference from 16 buffers. From a cost-effectiveness perspective, 2-3 transmit buffers are optimal. Average latency under saturation is independent of buffer count and grows linearly with fanout due to similar saturation throughput.

The full hybrid combining all three strategies uses 3 transmit queues. [Figure 16: see original paper] (64B) and [Figure 17: see original paper] (512B) show performance impacts of per-cycle arbitration count. For small packets, multi-arbitration combined with split transmission improves performance across all fanouts, increasing saturation throughput by up to 35 percentage points. Gains diminish with more arbitrations—4 vs. 6-8 arbitrations show only ~1 percentage point difference, making 4 arbitrations/cycle most cost-effective. Reduced serialization waiting time also decreases average latency, with diminishing returns beyond 4 arbitrations.

For larger packets (512B), multi-arbitration provides limited benefit (<1.5 percentage points maximum), being less effective than for small packets.

In summary, for multi-queue, multi-arbitration, split transmission structures, using 2-3 transmit queues with 3-4 arbitrations per cycle in split transmission mode offers the best cost-effectiveness, enabling various packet lengths to approach or achieve maximum saturation throughput at relatively low fanouts.

5 Conclusion and Future Work

This paper introduces the resource waiting graph concept to analyze multicast deadlock, theoretically proving necessary and sufficient conditions for deadlock-free multicast in virtual-cut-through multistage interconnection networks. Based on these conditions, we propose a centralized arbitration multicast engine with separated control and data flows, presenting multiple optimization strategies and design methods with detailed performance evaluations. The centralized arbitration structure using 3 transmit queues, 3-4 arbitrations per cycle, and split transmission offers excellent cost-effectiveness, achieving ideal multicast saturation throughput across a wide fanout range.

Future work includes building a prototype system based on this design to test actual multicast performance, introducing point-to-point reliable transmission mechanisms to ensure multicast reliability, and investigating efficient intra-node multicast for multi-processor systems and its coordination with inter-node multicast.

This project is supported by the 863 High-Tech Research and Development Program Project Dawning 5000A Efficient Computer (No. 2006AA01A102).

References

[1] John Kim, William J. Dally, Brian Towles, Amit K. Gupta. Microarchitecture of a High-Radix Router. in Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA ' 05), Madison, Wisconsin, USA, June 2005: 420-431

[2] G. Mora, J. Flich, J. Duato, E. Baydal, P. Lpez, O. Lysne. Towards an Efficient Switch Architecture for High-Radix Switches. ACM/IEEE Symposium

on Architectures for Networking and Communications Systems, December 2006

[3] S. Scott, D. Abts, J. Kim, W. Dally, "The BlackWidow High-Radix Clos Network," IEEE/ACM 33rd International Symposium on Computer Architecture, June 2006: 16-28

[4] C. Salvador. A Strategy for Efficient and Scalable Collective Communication in the Quadrics Network. Ph D Dissertation. University Politécnica de Valencia, 2005

[5] R. Riesen. Communication Patterns. Workshop on Communication Architecture for Clusters, Rhodes Island, Greece, April 2006

[6] D. K. Panda. Issues in Designing Efficient and Practical Algorithms for Collective Communication on wormhole-routed systems, ICPP Workshop on Challenges for Parallel Processing, August 1995

[7] Amith R Mamidala, Jiuxing Liu, D. K Panda. Efficient Barrier and Allreduce on Infiniband Clusters using Hardware Multicast and Adaptive Algorithms, Proceedings of the 2004 IEEE International Conference on Cluster Computing, San Diego, California, September 2004: 135-144

[8] J. Duato, S. Yalamanchili, and Lionel Ni. Interconnection Networks: an Engineering Approach. Morgan Kaufmann Publishers Inc., 2002

[9] Minghuang Guo, Ruayshiung Chang. Multicast ATM switches: survey and performance evaluation. ACM SIGCOMM Computer Communication Review, Volume 28, Issue 2, April 1998: 98-131

[10] Balaji Prabhakar, Nick McKeown, Ritesh Ahuja. Multicast Scheduling for Input-Queued Switches. in IEEE Journal on Selected Areas in Communications, Volume 15, Issue 5, June 1997

[11] Deng Pan, Yuanyuan Yang. FIFO Based Multicast Scheduling Algorithm for VOQ Packet Switches. In Proceedings of the 2004 International Conference on Parallel Processing (ICPP' 04), Montreal, Quebec, Canada, August 2004: 318 -325

[12] Lotfi Mhamdi, Mounir Hamdi. Scheduling multicast traffic in internally buffered crossbar switches. IEEE International Conference on Communications, Paris, France, June 2004

[13] Shutao Sun, Simin He, Yanfeng Zheng, Wen Gao. Multicast Scheduling in Buffered Crossbar Switches with Multiple Input Queues. 2005 Workshop on High Performance Switching and Routing, May 2005: 73-77

Author Biographies

Dawei Wang: Ph.D., High Performance Computer Research Center, Institute of Computing Technology, Chinese Academy of Sciences

Nongda Hu: Ph.D. Candidate, High Performance Computer Research Center, Institute of Computing Technology, Chinese Academy of Sciences

Note: Figure translations are in progress. See original paper for figures.