Hybrid Flow Control Methods

Eileen Peters, Steve Chou, Vicky Wong and Chuck Fraleigh

May 18, 1999

Abstract

Designing flow control mechanisms requires a trade off between latency, throughput, and cost/complexity. Circuit switching provides low latency but has very low throughput, while virtual cut-through flow control has better throughput but high latency. Wormhole flow control provides similar performance to virtual cut-through, but it requires much less buffer space. It is possible to create a hybrid flow control protocol which combines the best aspects of two or more of these basic flow control mechanisms. This paper will examine two proposed hybrid flow control mechanisms and evaluate their performance as compared to the basic flow control protocols. One hybrid combines virtual cut through and wormhole flow control, while the other combines circuit switching and wormhole flow control.

1 Introduction

This paper will investigate proposed methods for improving the latency and throughput of flow control mechanisms. As noted by [1], interconnection networks' throughput is typically limited to only 20-50% of the network capacity. This is because buffer and channel resources are allocated in a coupled fashion.

1.1 Existing Methods

Designing flow control mechanisms requires a trade off between latency, throughput, and cost/complexity. In table 1, the features of the dominant flow control methods are highlighted.

As shown in table 1, Circuit switching exhibits very low latency (once the circuit has been established). However, it provides very low throughput unless packet lengths are very long, i.e. transmission time is much longer than the time it takes to establish the circuit from end to end. Circuit switching offers the features that it is fairly simple to implement and does not require any buffers.

Store and forward flow control (SAF) has high latency and requires large buffers. However, it achieves higher throughput than circuit switching as it does not reserve the end to end channel for the entire duration of the packet transmission. Virtual cut-through flow control (VCT) has flits "cut-through" to the next node without waiting for the rest of the packet to arrive at the current node. This switching method has advantages and disadvantages similar to store and forward, but it achieves lower latency. The cost paid is that switches must be allocated on a flit-by-flit basis.

Wormhole flow control requires fewer, smaller buffers than VCT. At low traffic, wormhole has similar performance to VCT. However, for higher traffic, wormhole has higher latency and lower throughput due to contention for blocked channels. Blocked packets are not completely buffered, but instead hold upstream channels when packets are blocked. This achieves lower throughput. A performance improvement is to use wormhole flow control with virtual channels built on physical channels that ameliorate blocking problems. Wormhole with virtual channels improves throughput, but increases the cost/complexity and latency of packets.

1.2 Proposing Hybrid Methods

The existing flow control mechanisms have different balances of latency, throughput, and cost/complexity. Circuit switched packets consume more network resources while virtual cut-through flow control packets consume more node memory resources for buffering. Wormhole flow control strikes a balance between consuming network resources and node memory resources. It is possible to create a hybrid flow control protocol which combines the "good" aspects of two or more of these basic flow control mechanisms. Obviously, the hybrid flow control should have better performance than the basic protocols, or the effort to implement them is not worthwhile. We will examine three papers that propose hybrid flow control methods. We will measure the quality of proposed solutions based on their achieved latency and throughput, as well as their complexity and cost. These will be comparative measurements, based on the existing flow control methods.

This is a complex problem to solve. It is difficult to gain performance improvements in both latency and throughput simultaneously. In order to improve latency you typically need to reserve resources – buffers and channels (cir-

Flow Control Method	Latency	Throughput	Buffering Requirements
Circuit Switching	low	very low	none
Store and Forward (SAF)	high	medium	large
Virtual Cut-Through (VCT)	lower than SAF	medium	medium
Wormhole	lower than VCT	low	low
Wormhole w/ Virtual Channels	lower than VCT	medium	low

Table 1: Comparison of the dominant flow control methods

cuits, virtual channels) – which prevents other from using them. This has the effect of decreasing throughput. Thus reduced latency and increased throughput are difficult to achieve simultaneously. Another constraint in flow control problems is that when packets are blocked, they must consume either network resources (wormhole) or node memory resources (VCT). Hybrid methods of flow control will strive to find the optimum balance between these choices.

The key issues to be addressed are:

1. Which flow control mechanisms should be combined?
2. How do you select which flow control mechanism to use for each packet?
3. Does the proposed hybrid method perform better than the basic flow control mechanisms?

1.3 Background: Other Proposed Hybrids

Hybrid switching techniques that combine existing flow control methods have been proposed before. In [2], Buffered wormhole switching, pipelined circuit switching, and scouting switching are discussed.

Buffered wormhole switching combines wormhole and store and forward flow control. Short messages can be completely buffered as dynamically allocated chunks when a packet blocks. For such short messages, the input port can be released even though a blocked packet is there.

Pipelined circuit switching sets up the path before transmitting data with the header flit. However, unlike circuit switching, paths are formed by virtual channels. This combines circuit and wormhole switching, increasing resilience to component failures, but creating larger path setup times.

Scouting switching reduces the pipelined circuit switching overhead of path setup time. The first data flit does not wait until the head flit has set up the entire circuit. Rather, it must remain at least K links behind the routing header, where K is the "scouting distance." $K = 0$ is wormhole switching, and K very large is pipelined circuit switching. Thus, this hybrid method combines another hybrid method with wormhole switching. Basically it combines Circuit and wormhole switching, with an option of pure wormhole switching by modifying K values.

Analysis of these hybrids will not be performed in this paper, but are mentioned to give the reader an idea of the plethora of possible hybrid flow control methods.

1.4 The Proposed Hybrid Methods to be Examined

First, we will examine Hybrid Switching which combines virtual cut-through and wormhole switching [3]. Then, we will look at Wave Switching which blends wormhole and circuit switching, as well as wave pipelining [4][5].

2 Hybrid Switching

Hybrid Switching is proposed by Shin and Daniel in [3]. It combines wormhole and virtual cut-through flow control to achieve higher throughput and lower latency than wormhole switching can provide, while not requiring as much buffer space as virtual cut-through. The motivation is that networks that use wormhole switching saturate under lighter loads than networks that use virtual cut-through, due to contention for channels. On the other hand, wormhole switching significantly reduces the number and size of the buffers required when compared to virtual cut-through. Consequently, a network that can dynamically switch between the two flow control schemes would be able to respond to traffic load and achieve a better performance without a substantial increase in cost.

2.1 Mechanism and Implementation

Shin and Daniel have designed SPIDER (Scalable Point-to-point Interface DrivER), a communication interface that supports multiple switching schemes, to show the feasibility of Hybrid Switching and to evaluate its performance. A cycle-level discrete-event simulator is developed to accurately model the operation of SPIDER. It is used first to compare virtual cut-through and wormhole flow control in terms of average latency, the memory resources required and the maximum achievable throughput. They then evaluated the performance of Hybrid Switching using the same metrics. Table 2 shows a summary of the results.

Average latency	Lower than wormhole under heavy load
Memory required	Less than VCT under all load
Max. achievable throughput	Higher than wormhole for distance > h

Table 2: Performance of Hybrid Switching

Hybrid Switching balances the use of network and memory resources for storing blocked packets by using a field in the routing header. This field identifies the number of links a packet can hold while stalling the network. It is set to an initial value (h) at the source node and is decremented at each hop until h reaches 0. When a packet blocks, the router examines the value of h and buffers the packet if $h = 0$. Once a packet is buffered, h is reset to the initial value. Virtual cut-through flow control can be effectively implemented as Hybrid Switching with $h = 0$, while wormhole flow control is equivalent to Hybrid Switching with $h = \infty$. This scheme limits the number of links a blocked packet can hold and thus relieves some of the channel contention under heavy load. The initial value of h can also be changed dynamically, depending on the current load and the needs of the applications.

2.2 Strong Points

The motivation of implementing Hybrid Switching is well justified by the results of the experiments that compared wormhole flow control with virtual cut-through flow control. It is shown that wormhole reaches saturation under lighter load than virtual cut-through, which leads to a drastic increase in the mean packet latency. The maximum achievable throughput of wormhole also decreases as the distance travelled by packets increases, while virtual cut-through does not exhibit the same behavior. However, under virtual cut-through, the average rate of packets buffering at intermediate nodes increases as the load increases, and eventually surpasses the packet generation rate. This implies a severe memory resources requirement.

The evaluation of Hybrid Switching shows that it is beneficial to buffer blocked packets that are stalling several links. Hybrid Switching achieves higher throughput than wormhole, requires less storage resources than virtual cut-through, and its maximum achievable throughput increases with the packet distance (for distances greater than h). Therefore, it provides a good middle ground between wormhole and virtual cut-through flow control.

Furthermore, the use of SPIDER provides a common base for performance evaluation and allows fair comparison of the different schemes.

2.3 Weak Points

One way to increase the throughput of wormhole flow control is to use virtual channels to alleviate blocking [1]. This alternative is not considered in this paper. Both Hybrid Switching and virtual channel flow control improve upon wormhole and virtual cut-through. Without a comparison of Hybrid Switching against virtual channel flow control, it is not completely convincing that Hybrid Switching is an efficient flow control scheme.

SPIDER uses a central buffer memory that is shared among all injection and reception ports. While this allows the sharing of the memory bandwidth, the additional interface increases the time it takes to buffer a packet and thus prevents certain links to be freed up earlier. It may be better to provide a number of flit buffers at each port and only use memory when needed.

The number of buffers at each node is not limited in the simulations performed. This will not be the case in actual implementation. A more realistic simulation would have a fixed number of buffers. This could have an effect on the sustained throughput achievable by the hybrid flow control scheme.

2.4 What Remains To Be Done

The hybrid algorithm used in the paper is based on the threshold h that affects when to stall or buffer a blocked packet. Performance of this scheme using $h = 1$ and $h = 2$ has been shown, but the optimal value of h is yet to be determined. It is also unclear what factors should determine h. Some parameters mentioned in the paper are buffer space available, distance to be traversed, and how long a packet is blocked. Further study can be done to refine the algorithm.

As mentioned in section 2.3, studies comparing Hybrid Switching and virtual channel flow control can be done to see how efficient this scheme is. Moreover, simulations with fixed number of buffers or experiments with physical router chips would provide more accurate results for the performance evaluation.

3 Wave Switching

Wave switching is proposed by Lopez, Duato, Silla and Yalamanchili in [4]. It combines wormhole switching and circuit switching within the same router architecture to achieve very high performance by taking advantage of communication locality. For example, if two nodes communicate frequently with one another, a physical circuit is established between them, whereas for nodes that communicate infrequently, regular wormhole switching is used. All physical circuits are handled in the network through the use of a history table of paths (similar to a cache) so circuits can be established without the usual path finding overhead. Furthermore, when two nodes are going to exchange a large amount of information, the compiler can generate instructions that instruct the router to set up the circuit before the circuit is even needed. This concept is similar to memory and instruction prefetching in a microprocessor. By combining wormhole and circuit switching together efficiently, network bandwidth can be increased significantly while also reducing latency for communications that use pre-established physical circuits.

3.1 Principles Behind the Architecture

In the design of distributed memory multiprocessing systems, there are inherent advantages and disadvantages in choosing a circuit switched or wormhole switched architecture. Wormhole flow control is attractive because it is simple and fast, offering fairly low latencies compared to other flow control schemes. However, inherent in all wormhole schemes is the disadvantage that contention causes messages to remain in the network, potentially preventing the use of the channels they occupy and wasting channel bandwidth. These contention problems can be alleviated through the use of virtual channels and adaptive routing algorithms at the expense of a more complex router with a higher node delay. This increase in complexity ultimately leads to a higher latency of the network.

Circuit switching on the other hand is extremely effective in reducing latency when network traffic exhibits spatial and temporal locality. Once a circuit is successfully set up, there is no contention for the messages using that circuit resulting in an extremely high bandwidth connection between two nodes. The limitation of circuit switching however, is that there are limited resources within the network. If a new circuit requires the use of certain channels belonging to an established circuit, the established circuit must be torn down to accommodate the new path. Furthermore, in traditional circuit switching schemes, the routing path needs to be set up ahead of time resulting in a lower throughput. In general, circuit switching is only beneficial when two nodes are going to exchange large amounts of information because the increased bandwidth makes up for the latency of circuit establishment.

Wave switching combines the best aspects of both switching schemes by executing both switching schemes concurrently. For nodes that communicate frequently with each other and transmit long messages, circuit switching is used to maximize bandwidth. By using wave pipelining within circuit switching, extremely high clock frequencies can be used to achieve up to 4 times the bandwidth of wormhole routers of the same technology. All other transactions that are small (<128 flits) will use wormhole flow control through the network. By combining wormhole and circuit switching, we can achieve high bandwidth with extremely low latencies.

3.2 Implementation

Two protocols are used to implement wave switching. The first protocol handles the caching of circuits, automatically establishing a circuit when two nodes are going to communicate. When a new circuit requires the resources of an existing circuit, a replacement algorithm selects which circuit will be torn down. The second protocol handles when a circuit should be established or torn down and is usually implemented by the programmer and/or the compiler.

From an architectural standpoint, pre-established circuits require the use of dedicated physical channels to form a path between two nodes. In wave switching, physical circuits are set up and torn down by using a set of dedicated virtual channels. The wormhole aspect of wave switching, uses another set of independent virtual channels. The router consists of several switches $s0$, $s1$, $s2$, etc., and two routing control units. One control unit along with switch $s0$ implements wormhole switching while the remaining switches are used to implement circuit switching. (See figure 1) With the $s0$ switch, physical channels are split into $k + w$ virtual channels. Of these, k channels are used as control channels to set up and tear down physical circuits. The remaining w virtual channels are used to transmit messages using wormhole switching.

Looking at the wormhole aspect of wave switching, messages are routed using either a deterministic or adaptive routing algorithm that is deadlock free. The circuit is established by sending a probe through the control channels. The probe traverses the network forming a path until the destination node is reached. Once the destination is reached, an acknowledgement is sent back to the source. To maximize the likelihood of establishing a circuit, a misrouting backtracking algorithm with a finite number of misroutes is used. A cache or "History Store" of registers keeps track of all paths already searched by each probe so they can be reformed quickly and to avoid the searching of the same path twice.

3.3 Routing Protocols

The routing protocol for wave switching networks must decide whether to use physical circuits or wormhole flow control for each packet that is transmitted. Circuits are effective for long messages or when multiple messages need to be sent between two nodes (i.e. when there is temporal locality). Wormhole flow control performs well for short messages. Two routing protocols, Cache-Like Routing Protocol (CLRP) and Compiler Aided Routing Protocol (CARP), are presented in [5].

CLRP attempts to use physical circuits for every packet that is transmitted. When a packet is transmitted, CLRP first checks to see if a circuit has already been established between the source and destination. If a circuit exists, it is used to transmit the packet. After the packet has been transmitted, the circuit remains connected in order to handle future traffic between the two nodes. If a circuit does not exist, then a routing probe message is sent to the destination to try to set up a circuit. This probe can backtrack and misroute a fixed number of times when attempting to create the circuit. If a circuit cannot be created, a second routing probe is transmitted to the destination. This probe has a Force bit set which causes intermediate routers to tear down existing circuits if no traffic is being transmitted on the circuits. If the second probe fails, the packet is transmitted using wormhole flow control.

CLRP can be viewed as cache protocol for virtual circuits. If there is a cache hit, the protocol uses the circuit that is already established. If there is a miss, CLRP attempts to use a circuit which is free. If all the circuits are busy, it selects one circuit to replace.

The second proposed protocol, CARP, determines at compile time if a physical circuit should be established to transmit a packet or if the packet should be transmitted using wormhole flow control . The compiler can determine how many packets will be sent between two nodes and set up a circuit if the number is larger than a certain threshold. At run time, a special instruction causes a circuit to be established between a specific source and destination using the routing probes as described above. Note that the circuit may be established before the first message needs to be transmitted eliminating the overhead for circuit setup. When all of the messages have been transmitted a special instruction is used to tear down the circuit.

3.4 Strong Points

The architecture of the wave switch router effectively combines circuit switching and wormhole flow control. Physical circuits are very effective when there is a lot of data to transmit between two nodes. Using circuits when there is a lot of data to be transmitted and using wormhole flow control for small messages should improve the performance of the network. Using a compiler to establish the circuits before the message needs to be transmitted will further improve performance by eliminating the routing overhead.

The wave switch router also successfully increases throughput by using wave pipelining for the physical circuits. As was shown in [4], wave pipelined circuits can achieve high throughput and low latency when the overhead for circuit establishment is ignored, as would be the case when using the CARP protocol.

3.5 Weak Points

In order to be effective, wave pipelining routers must make an appropriate choice between using physical circuits and using wormhole flow control. It is unclear if the CLRP or CARP protocol can effectively make this choice. CLRP always attempts to create a circuit for every packet that is transmitted. This could result in a large amount of overhead and very long latencies if the circuits cannot be reused (i.e. if there are not many cache hits). CLRP could be improved by looking at the packet size or other information when trying to determine if a circuit should be established for a packet. CARP assumes the circuits can be setup before a packet needs to be transmitted. As shown in [4], this can result in very high throughput and low latency. However, it is unclear how much locality of reference exists in interprocessor communication. If there is not much locality, then the compiler would only rarely create circuits, and the extra effort of implementing the circuit switching protocol may not be justified.

3.6 What remains to be done

Additional performance analysis of the wave switch architecture still needs to be done. Some analysis had been performed assuming no overhead for circuit setup. More realistic analysis using the CLRP and CARP protocol should be performed, and the results should be compared to a wormhole router using virtual channels.

The development of a compiler which supports CARP also needs to be developed. This is necessary in order to determine what criteria should be used when determining if physical circuits or wormhole flow control should be used.

4 Critique of Proposed Hybrid Flow Control Methods

Improving the performance of flow control is an important problem to solve in order to the throughput of interconnection networks. Different types of flow control work better in different situations (e.g. long packets perform well with circuit switching, while short messages obtain better throughput with wormhole flow control). However, it remains to be seen if the overhead of implementing one of these hybrid methods is offset by performance gains.

4.1 Criticism of Methods

The papers are addressing many of the critical issues of hybrid flow control, but have missed some important ones. First of all, performance analysis and results are minimal. The hybrid flow control methods add a lot of complexity but they do not show that they gain worthwhile improvements in latency or throughput. Without adequate analysis, it is impossible to determine the extent of benefits gained. While both hybrid proposals perform better than traditional wormhole flow control, none of the papers compares the proposed solution with wormhole flow control with virtual channels.

Table 3 summarizes the relative performance of Hybrid Switching and wave switching compared with wormhole flow control. However, enough research needs to be done to generate data and performance results, such that the table 3 can be completed in comparison with other methods that achieve a balance of cost/complexity, latency and throughput as well as network and node memory resources. Specifically, a comparison should be done with wormhole flow control with virtual channels.

Another issue is that neither paper adequately addresses how to select which flow control mechanism to choose for each packet. Hybrid Switching just does something simple (it uses a value for h) to determine the balance between VCT and wormhole flow control. However, the paper does not address the issue of what value of h should be used or even how this value should be selected. Wave switching does propose two routing mechanisms which can be used to select either circuits or wormhole, but the papers do not show any performance analysis.

Thus, we recommend a focus on the important question of how to select which flow control mechanism to use for each packet. Will this be done by the hardware at runtime or by the compiler, or a combination? Significant compiler research needs to be done to exploit the benefits of a hybrid switching technique. If the right choice of flow control methods is not made for each packet, there is no gain from supporting two or more choices. Assuming all the flow control mechanisms can be implemented in a router, experimentation can be done to see if there is a benefit to choosing between the different flow control methods. Furthermore, it would also see if the compiler or hardware can make choices that significantly improve performance based on packet length, as well as temporal and spatial locality of circuit use.

4.2 Alternative Approach

These proposed hybrid methods are basically trying to improve upon wormhole flow control by combining it with another existing method. Perhaps instead of trying to combine existing flow control schemes, we should improve the current ones or invent new ones. For example, flit reservation flow control is an improvement upon virtual channel flow control which adds special headers and special resources allocation.

References

[1] W. J. Dally, "Virtual-Channel Flow Control," IEEE Trans. Parallel and Distributed Systems, vol. 3, pp. 194-205, March 1992.

[2] J. Duato, S. Yalamanchili, L. Ni, "Interconnection Networks: An Engineering Approach," IEEE Computer Society, 1997, Chapter 2, Section 2.5, pp. 5-60.

[3] K. G. Shin and S. W. Daniel, "Analysis and Implementation of Hybrid Switching," in Proc. International Symposium on Computer Architecture, pp. 211-219, June 1995.

[4] J. Duato, P. Lopez, F. Silla and S. Yalamanchili, "A High Perfomance Router Architecture for Interconnection Networks," in Proc. International Conference on Parallel Processing, August 1996.

[5] J. Duato, P. Lopez and S. Yalamanchili, "Deadlock and Livelock Free Routing Protocols for Wave Switching," in Proc. International Parallel Processing Symposium, April 1997.

Flow Control Method	Latency	Throughput	Buffering Requirements
Hybrid Switching	Better than wormhole	Better than wormhole	Less than VCT
Wave Switching	Better than wormhole	Better than wormhole	Better than wormhole

Table 3: Properties of Hybrid and Wave Switching

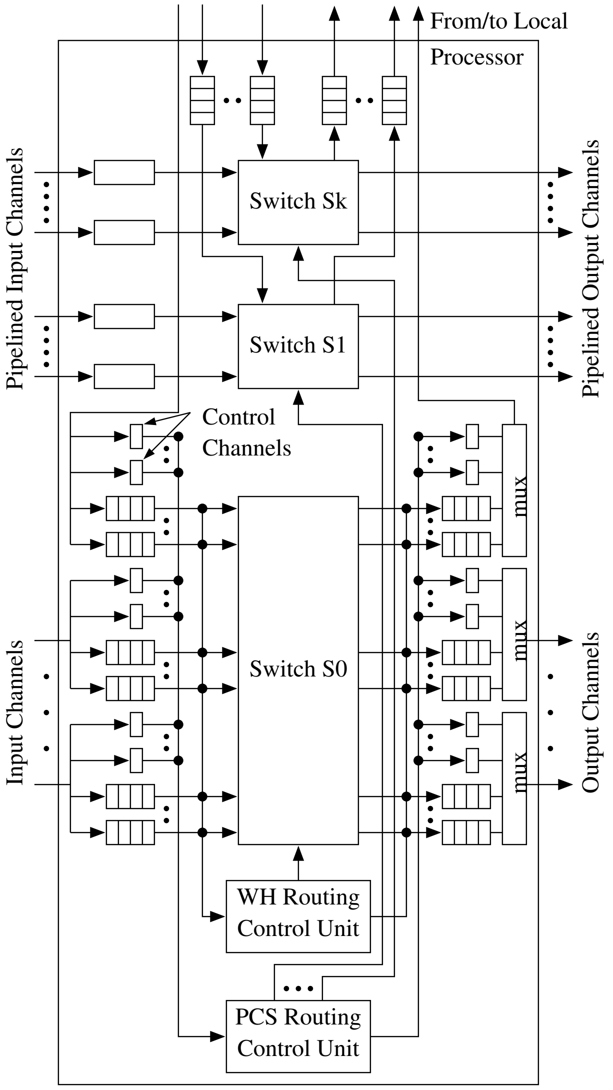

Figure 1: Router Architecture