Architecture and Hardware Support for Real-time Scheduling of Packet Streams

Raj Krishnamurthy, Sudhakar Yalamanchili, Richard West, and Karsten Schwan
{rk@cc.gatech.edu}

Critical Systems Laboratory
Georgia Institute of Technology
Atlanta, GA 30332

The Problem

1) Scheduling of packet streams in real-time (as opposed to virtual-time) is necessary to make classes of scheduling guarantees and maximize link utilization.
2) Scheduling at wire-speeds for optical multi-gigabit links and the emerging 10GEA (10 Gigabit Ethernet Alliance) standard necessitates scheduling decisions be guaranteed to be completed in a packet time.
3) Architecture and implementations that can meet cost/performance requirements across a range of environments without ASIC re-engineering overheads.
4) It is necessary to trade-off scheduler throughput in packets/sec with quality of service and scheduling granularity, e.g., scheduling at the packet level vs. MPEG frame level.

Proposed Solution

We propose to address these problems through the use of a powerful scheduling discipline and a flexible target architecture that combines a commercial microprocessor datapath with a tightly coupled reconfigurable logic component such as a FPGA. Such system on a chip architectures have been announced in the recent past and provide potential hardware solutions for applications that demand both flexibility and the performance that can be achieved via hardware customization. Our approach implements the compute intensive scheduling decision logic within the configurable logic component while control and data movement is handled by the microprocessor [2].

The scheduling discipline for which we propose hardware solutions is Dynamic Window-Constrained Scheduling (DWCS) [1, 2]. DWCS is a powerful scheduling framework that can be configured to implement most existing scheduling disciplines such as WFQ [3]. While DWCS addresses the issue of provisioning QoS, the complexity of the priority update computations poses a challenging implementation problem for scheduling a large number of streams over multi-gigabit links. For example, the ethernet frame time on a 10 Gigabit link ranges from approximately 0.05 microseconds (64 byte) to 1.2 microsecond (1500 byte). This can be substantially lower for ATM cells or SONET frames that need to be scheduled at wire speeds. Packet level QoS scheduling at these link speeds poses significant implementation challenges.

To meet the challenge we propose a scheduler architecture comprised of a microprocessor coupled with a field programmable gate array (FPGA). Scheduling logic does possess significant amount of parallelism for which we propose a customized FPGA solution. Such solutions are viable as FPGA technology pushes 10 M gate designs with clock rates of up to 200MHz with relatively low reconfiguration overheads. By carefully crafting suitable implementations for compute intensive scheduler components for implementation within the FPGA, we find tractable implementations for the fine grained, real-time packet scheduling problem.

The Scheduler: Operation

The central idea in any packet scheduling discipline is to find the winner stream among any two streams based on a set of rules. Pairwise ordering of streams produces an overall winner. DWCS uses deadlines and loss-tolerances (x packets in a y packets interval can be late/lost) to capture the notion of priority. In DWCS packet deadlines are used to schedule two streams. If

the deadlines are the same then, a set of rules based on loss-tolerances is used to determine the winner. Once a winner is computed, the packet at the head of this stream can be transmitted and the priorities of other streams waiting to be serviced is adjusted based on relative importance and whether deadlines have been missed. This process is repeated until all the packets waiting to be serviced are scheduled. It is key to note that a winner must be determined and priorities of other streams adjusted before the next winner can be computed (since loss-tolerance values are adjusted at the end of each scheduler cycle). The implication is successive scheduling operations cannot be overlapped/pipelined.

The Scheduler: Hardware Architecture

The hardware implementation consists of a two basic components. The *register block* holds the state of each stream including the priority values (deadlines and loss-tolerances) and priority update logic. Register blocks supply priority values to a decision block. The decision block is organized as a set of components that execute concurrently – a comparator for comparing the priority values and one component for each rule used by the scheduler in case ties need to be broken. This decision speed is dictated by the critical path through the most complex component. Two register blocks feeding a decision block is a *scheduling tile* and is considered a base structure. Preliminary performance values for a scheduling tile are shown in Table 1 below:

Table 1: Implementation Results(Synthesis from VHDL and Xilinx Core Library)

Architecture	Implementation	Number of CLBs	Maximum Clock
Canonical Structure (Two register blocks feeding a decision block)	Xilinx Virtex V300BG432-6. Synthesis optimization set at "low" . Clock target set at 100MHz.	7 %	64.65 MHz

Implementation results detail the area (number of Configurable Logic Blocks) and maximum clock period attainable by this design. The initial results are very promising. For four streams, a winner can be computed in 2 clock cycles or every 30 ns with another cycle for priority update or every 45ns(65 MHz, 15ns period). The logic utilization is also very low, close to 10 % of a Virtex 300 part for a single tile. A number of Decision Blocks may be placed on a Virtex part for scheduling of multiple streams.

For scheduling a large number of streams, we are investigating an architectural solution based on a single stage, re-circulating shuffle interconnection. Register blocks are connected to a stage of two input decision blocks via a shuffle connection. The outputs of the decision blocks are fed back to the inputs. The shuffle architecture for a four stream version consists of four register base blocks and two decision blocks. The process takes 2 (log4) cycles to complete and provides a priority ordered list of streams. The highest priority stream is selected and priorities updated accordingly by the base register blocks. Thus for N streams, we require N register blocks, N/2 decision blocks and logN cycles to pick a winner. We also require control and steering logic unit that provides interface to memory, winner comparator block outputs and buses to provide stream parameter values to the register base blocks . The logic requirements are approximately 50% of that of a typical comparator tree implementation. Note that the scheduler area grows linearly in the number of streams and scheduling decision time delay grows logarithmically.

The scheduling tile is designed to be flexible enough to be implement alternative scheduling algorithms. Performance results include area/time performance as a function of the number of streams.

References

[1] Richard West and C. Poellabauer. Analysis of a Window-Constrained Scheduler for Real-time and Best-Effort Packet Streams. In *Proceedings of the 21st Real-Time Systems Symposium, Orlando, Florida, November 2000.*

[2] Raj Krishnamurthy, K. Schwan, R. West and M. Rosu. A Network CoProcessor-Based Approach to Scalable Media Streaming in Servers, In *Proceedings of the 29th International Conference on Parallel Processing, Toronto, Canada, July 2000.*

[3] A. Demers, S. Keshav and S. Shenker. Analysis and Simulation of a Fair-Queueing Algorithm. *Journal of Internernetworking Research and Experience, pages 3-26, Oct 1990*