Wormhole IP over (Connectionless) ATM

Manolis Katevenis, Iakovos Mavroidis, Ioannis Mavroidis, and Georgios Glykopoulos
University of Crete and FORTH
ICS-FORTH, P.O.Box 1385, Heraklion, Crete, GR 711 10 Greece
E-mail: katevenis@ics.forth.gr Tel.: +30 (81) 391664 Fax: +30 (81) 391661

http://archvlsi.ics.forth.gr/wormholeIP/ – version: July 1998

ABSTRACT: In the eighties, high throughput and low latency requirements in multiprocessor interconnection networks led to wormhole routing. Today, the same techniques are applicable to routing internet packets over ATM hardware at high speed. Just like virtual channels in wormhole routing carry packets segmented into flits, a number of hardware-managed VC's in ATM can carry IP packets segmented into cells according to AAL-5; each VC is dedicated to one packet for the duration of that packet, and is afterwards reassigned to another packet, in hardware. This idea was introduced by Barnett [Barn97] and was named connectionless ATM. We modify the Barnett proposal to make it applicable to *existing* ATM equipment: we propose a single-input, single-output *Wormhole IP Router,* that functions as a VP/VC translation filter between ATM subnetworks; fast IP routing lookups can be as in [GuLK98]. Based on actual internet traces, we show by simulation that a few tens of hardware-managed VC's per outgoing VP suffice for all but 10^{-4} or less of the packets. We analyze the hardware cost of a wormhole IP routing filter, and show that it can be built at low cost: 10 off-the-shelf chips will do for 622 Mb/s operation; using pipelining, operation is feasible even at 10 Gb/s, today.

KEYWORDS: *IP over ATM, connectionless ATM, wormhole routing, gigabit router, wormhole IP, routing filter.*

1. Introduction

The global internetwork of the future requires high throughput, low latency, quality guarantees, and the coexistence of various traffic types –data, voice, video, and perhaps more. An important part of this traffic (at least data) comes in *variable-size packets* and is connectionless or requires *no* (appreciable) *delay* for connection set-up. The internet protocol (IP), which is currently used for global internetworking, is connectionless and uses variable-size packets.

Fast switching hardware, on the other hand, strongly favors *small, fixed-size units* of traffic, and fast routing decisions e.g. through small connection tables. Fixed-size units are preferred for a number of reasons. First, in general, they simplify hardware, thus reducing its cost and increasing its speed; it is precisely this effect that led RISC processors to adopt fixed-size instructions rather than the variable-size instructions found in earlier, CISC machines. Second, implementing multiple queues (e.g. per-class, per-output, per-flow) inside a shared buffer memory, in hardware, is almost impossible unless all memory allocation is done in multiples of a fixed-size block unit. Third, efficiently scheduling the traffic over a crossbar switch is almost impossible unless all traffic sources start and finish their transmissions in synchrony, thus implying that they all use a common-size unit of transmission. The fixed-size unit of traffic must be relatively *small,* so as to reduce delay for high-priority traffic. The upper bound on delay that can be guaranteed for high-class traffic is usually equal to a certain maximum number of low-class traffic units that high-class traffic may encounter in front of it, times the delay caused by each such unit; the smaller the size of the traffic units, the shorter this upper bound on delay becomes[1].

How can these hardware preferences be reconciled with the application requirements for variable-size packets and connectionless routing? Multiprocessor technology has already done that, in the eighties, using *Wormhole Routing* [DaSe87], [Bork90], [Gall97]. Figure 1 shows the basic idea of virtual-channel (multilane) wormhole routing [Dally92]. Variable-length packets are segmented into fixed-size *flits* (flow units); flit sizes have usually ranged from 4 to 16 bytes. A (modest) number of *virtual channels* (VC) is defined on each physical link. All flits of a given packet are transmitted on a single VC. Each flit has a header identifying the VC that the flit belongs to. Additionally, each flit identifies itself as a head, middle, or tail flit. A head flit is the first flit of a packet, and carries the packet's destination address; a tail flit terminates its packet and frees its virtual channel. Free virtual channels are reassigned to new packets. A wormhole router maintains a mapping table of incoming link-VC pairs onto outgoing link-VC pairs; this table is consulted for middle and tail flits, with tail flits deactivating their mapping entry. Head flits consult a routing table, in order for the proper outgoing link(s) to be determined; then, a free VC is sought on this (these) link(s), and a corresponding mapping entry is established.

Asynchronous Transfer Mode (ATM) [LeBo92] is a networking technology that was developed in the last ten years with the intention of providing a common infrastructure for networks carrying all traffic types; there is considerable deployment of ATM hardware and software today, in

[1] additionally, small traffic units minimize the percentage of resources that remain unused due to fragmentation; of course, we do not want the traffic units to be too small, because that increases the header overhead and the switch controller rate or operation.

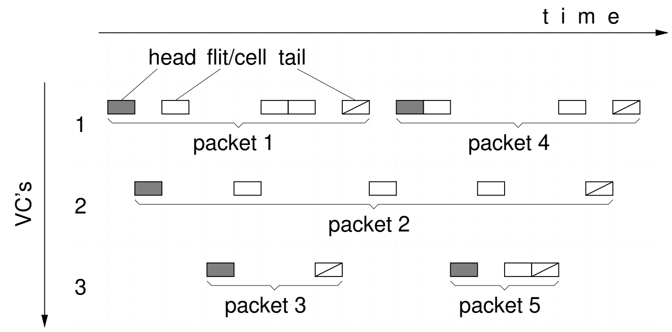

Figure 1: Packets segmented into flits or cells and interleaved on multiple VC's

local and wide area networks. ATM features the desirable hardware characteristics listed above: traffic is carried in fixed-size units called *cells,* and the cell size is reasonably small: 48 bytes of payload plus 5 bytes of header. Cell routing decisions are simplified by the presence of a connection table, indexed using a number of bits from the cell header. These index bits can be as many as 28, leading to large (and slow) tables; on the other hand, in many practical cases, it suffices to look at a lot fewer bits from the header (e.g. VP switching), thus leading to small and fast switches.

There are notable analogies between ATM and wormhole routing. In an ATM network, variable size packets are fragmented into cells before entering the network. Cells are transmitted over virtual circuits (VC) and virtual paths (VP), which are fixed routes over the network. All cells of a packet are assigned the same VC and are transmitted in order. This bears similarities to the operation of wormhole networks, as presented in figure 1: ATM cells are analogous to wormhole flits.

Connectionless ATM has been proposed by Barnett as a method for fast routing of IP packets over ATM hardware [Barn97]. Although Barnett does not mention wormhole routing, his proposal is essentially identical to adapting a wormhole router to operate on IP packets that have been segmented into ATM cells according to ATM Adaptation Layer 5 (AAL5). Because of this similarity, we propose and use the name *Wormhole IP over ATM* for what Barnett called Connectionless ATM. Wormhole routers, of course, also use backpressure (credit-based flow control), while ATM (usually) does not. The rest of this paper assumes conventional ATM technology, which does not use credit-based flow control, i.e. there is a difference in that respect from wormhole routing. Notice, however, that backpressure can be advantageously used with ATM too [KaSS97].

Wormhole IP over (connectionless) ATM is a hardware-only IP routing method, and hence differs fundamentally from the software methods that are in use today for routing IP over ATM

–or for using ATM under IP [NeML98]. Software methods have a number of costs, overheads and associated delays that are not present in wormhole IP over ATM: *(i)* a very large number of VP/VC labels are consumed, or *(ii)* until a flow has been recognized, the first few packets in the flow are not switched; and *(iii)* new connections for new flows are established in software. These are further discussed in section 5.

This paper, first, modifies the Barnett proposal so as to make it practically realizable at low cost: instead of using *modified* ATM switches (CATM/ACS switches in [Barn97]), we propose to only build *single-input, single-output* wormhole IP routers, that function as VP/VC translation filters and interoperate with *existing* ATM switches and networks (section 2). Second, we simulate the operation of the proposed routing filter using actual internet traces, and we study its performance (packet loss probability when bufferless, else delay and buffer space) as a function of various parameters (number of VC's, network load, policing). The results are presented in section 3 and are very optimistic: a few tens of VC identifiers per outgoing VP suffice. Third, in section 4, we analyze the hardware structure of such a wormhole IP routing filter, and show that its cost is low, especially if it is made bufferless; operation at multi-gigabit per second rates is feasible, today.

2. Wormhole IP Routing Filter

Figure 2 illustrates the generic operating environment of the wormhole IP over ATM router proposed and studied in this paper. To reduce its cost and to maximize its flexibility, it has a single incoming and a single outgoing link, thus operating as a *filter.* The basic idea for this routing filter is to interoperate with *existing, conventional* ATM equipment, and to route IP packets by selecting the (permanent) *virtual path (VP)* through an ATM subnetwork that each packet is forwarded on. The routing filter operates on IP packets that have been segmented according to AAL5, *without reassembling* them, thus avoiding the delay and complexity of segmentation and reassembly (SAR) at each hop through a series of ATM subnetworks. The proposed wormhole IP router handles a *mixed traffic* consisting of native ATM connections and IP packets transported over ATM.

The proposed wormhole IP over ATM router operates as a filter between two *ATM subnetworks.* This general formulation encompasses a number of interesting cases:

- *Entry point* into an ATM network. The ATM subnetwork feeding the filter may be null; thus, the routing filter immediately follows an IP source or a device that receives IP packets from a non-ATM network and segments them into ATM cells according to AAL5.

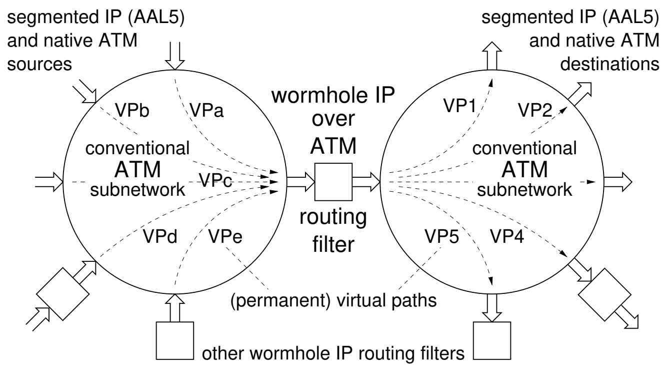

Figure 2: Wormhole IP router as a filter between two conventional ATM subnetworks

- *Exit point* from an ATM network. The ATM subnetwork following the routing filter may be null. In this case it is interesting to use a routing filter configured with a single outgoing VP and VC, which will have the effect of reassembling ATM cells into IP packets, so that these can be handed to an IP destination or to a non-ATM network.
- *Single switch.* An ATM subnetwork may consist of a single ATM switch; an ATM switch with wormhole IP routing filters attached to all its inputs is similar to a connectionless ATM (CATM) switch in the Barnett proposal [Barn97]. If we were to build things from scratch, we could take such a wormhole IP switch/router as our building block, and optimize it to cost less than the sum of a conventional ATM switch plus several wormhole IP filters. With the installed basis of ATM equipment, however, today, the proposed filter forms a better building block, which has the additional advantage of being usable with arbitrarily-sized ATM (sub-) networks.
- *Medium-sized ATM subnetworks.* We can grow the subnetworks to which the routing filter connects progressively more and more. As we do so, the number of wormhole IP filters in the system is reduced, while the number of (permanent) VP's in each of these (fewer and larger) subnetworks grows. According to the tradeoff between these two factors, the network architect can choose to position the routing filters in the way he(she) considers best; the proposed filter is a flexible building block for all kinds of configurations. Note here the analogy between the (permanent) VP's in an ATM subnetwork here, and those in tag switching [RFC2105]. The difference of wormhole IP from tag switching is that with the wormhole approach no segmentation and reassembly (SAR) is needed at the interfaces between two ATM subnetworks.

In this paper we assume that the two links (input, output) of a wormhole IP routing filter are unidirectional; in this way, the rate of processing in the filter is kept low. The links could also be bidirectional; in that case, operation of the filter in one direction would be identical to and independent of its operation in the other direction.

2.1 *Operation of the Routing Filter*

A wormhole IP over ATM routing filter operates by looking at each ATM cell that passes through it, as shown in figure 3. For each incoming cell, we look up its virtual path (VP) and virtual circuit (VC) identifiers in the filter's *ATM Connection Table.* If the cell belongs to a native-ATM connection, it is forwarded to the output, possibly after undergoing VP/VC translation. Otherwise, under certain circumstances to be examined below, this may be the first cell of an IP packet, in which case it contains the IP destination address. That address must then be looked up in the *IP Routing Table.* This routing table specifies the desired output VP; an unused VC identifier in that VP is then requested from the *VC ID free list.* If no such unused VC currently exists, then the cells of this packet must be buffered in expectation of some VC getting freed, or else the packet can be dropped if such events are believed to occur rarely; thus, a wormhole IP routing filter may or may not contain *buffer memory* for cells.

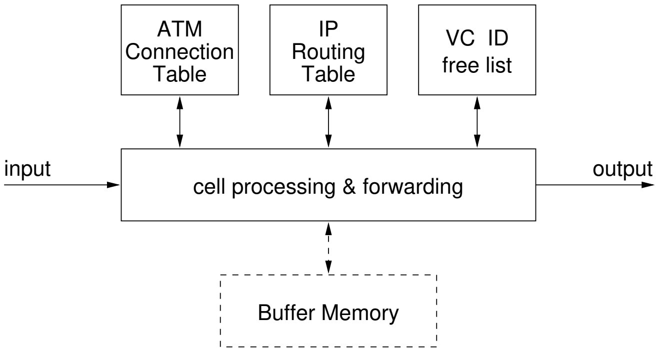

Figure 3: Functions in a wormhole IP routing filter

The IP Routing Table is the most challenging component of the filter –the other parts are relatively easier, as will be seen shortly. Fortunately, a lot of research was carried out recently on ways to speed up IP routing table look up [BCDP97], [WVTP97], [GuLK98]. The methods of [BCDP97] and [WVTP97] are rather software-oriented, while [GuLK98] is appropriate for hardware-only implementation. In this paper we use this latter method, whereby five 64 Mbit DRAM chips suffice for these IP route look-up's to be performed in a worst-case delay of two DRAM accesses each.

Figure 4 shows the ATM connection table in the routing filter of figure 3. The table is indexed using the VP and VC number of each incoming cell (less than the full 28 bits of these fields will suffice). For each such incoming VP/VC pair, the table lists a type (1 bit), a current state (2 bits), and an outgoing VP/VC pair (several bits, up to 28). The *type* bit differentiates the connections that have been set aside for native ATM traffic from those that the wormhole IP mechanism "owns" and uses. The VP/VC pairs of *type ATM* are owned and managed by the ATM network manager; they are used exactly like conventional connection identifiers, to label native-ATM connections. The wormhole IP hardware never uses these VP/VC pairs.

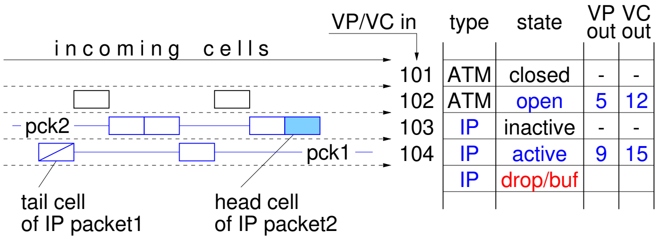

Figure 4: ATM connection table contents

The VP/VC identifiers of *type IP* are managed by the routing filter, in hardware. The ATM network manager considers that all IP-type VP/VC pairs correspond to *legal, open* ATM connections, that are unavailable for transporting other (native-ATM) traffic. As shown in figure 2, the ATM network switches these (IP-type) VP/VC pairs based on their VP value (VP switching)[2] (the same can be done for the ATM-type pairs, but this is immaterial for the operation of the wormhole filter). While the ATM manager considers all IP-type VP/VC pairs as open ATM connections, the wormhole IP hardware differentiates them into *active* or *inactive,* using the *state* field of the connection table. An active VP/VC pair is one on which we have started receiving the cells of an IP packet, while not all cells of that packet have yet been received (packet 1 in figure 4). An inactive VP/VC pair (of type IP) is one on which all previously received IP packets have been received in full, and no new packet has arrived yet (packet 2 in figure 4).

The operation of the wormhole IP over ATM routing filter is described by the flow diagram of figure 5. For each incoming cell, the filter looks at its VP/VC number. One simple option (*not* the one shown in figure 4) is to distinguish ATM-type connections from IP-type ones based on the VP/VC field matching some particular bit pattern(s). Under this option, cells belonging to ATM-type connections are immediately and "blindly" forwarded to the output, without any processing. Figures 4 and 5 show the other option: the connection table contains entries for ATM-type VP/VC's, too. These entries *(i)* allow fine granularity in choosing the type of each VP/VC pair (minor point), and *(ii)* contain a translated VP/VC pair (major point), along with a validity bit (open/closed state). Under this option, shown in a dashed box in figure 5, the routing filter provides VP/VC translation for native-ATM traffic. This option may be useful for the following

[2] of course, as far as the wormhole IP operation is concerned, *any subfield* of the 28-bit VP/VC field can be used as what we call "VP", and any other subfield can be used as "VC" –the choice is up to the ATM network manager.

reasons. First, the routing filter already contains large (DRAM) memories, and the inclusion of a (quite) larger-than-necessary connection table may often incur only a small incremental cost. Second, relatively small ATM subnetworks can be configured to operate without any internal VP/VC translation; on the other hand, when joining together such subnetworks to form a large ATM network, VP/VC translation becomes necessary, and the translation table must now be large. As illustrated in figure 2, the proposed routing filter will often be positioned at precisely the locations where native-ATM prefers to have its (large) VP/VC translation tables; we then have the option of simplifying the ATM subnetworks, by not requiring them to perform any translation, while including such translation in the wormhole IP filters, where it will often come for only a small incremental cost.

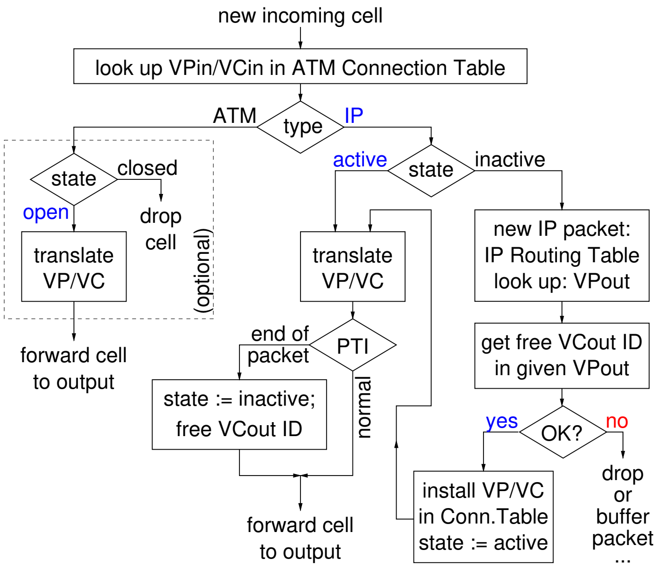

Figure 5: Flow diagram of filter actions per cell

When the incoming cell is of type IP, the state of its VP/VCin pair is examined. If the cell arrived on an *active* VC, it means that a (temporary, wormhole) connection has already been set up for the cells of the packet that it belongs to. The connection table specifies the VP/VC translation that has been established for these cells. After translation, if this is not the last cell of the packet, we merely forward it to the output and we are done with it. Remember that, under AAL5 segmentation, the last cell of a packet is marked using one of the bits in the *Payload Type Indicator (PTI)* field of the cell header. Now, if this PTI field of the cell indicates that this is the last cell of its IP packet, the temporary, wormhole connection that had been established for this packet has to be torn down; this is accomplished by marking its state as inactive in the connection table, and by returning the VPout/VCout identifier to the free list.

If the incoming cell arrived on an *inactive* VP/VC pair of type IP, then it is the first cell of an IP packet, since the connection became inactive either by initialization or by the tail cell of another IP packet (the effects of cell loss in the ATM subnetworks will be examined in section 2.2). The destination address of an IP packet that has been segmented according to AAL5, either without or with multiprotocol encapsulation, is fully contained in the first ATM cell of the segmented packet (this will also be true for IPv6). Taking advantage of this fact, the wormhole IP routing filter extracts this IP destination address from the cell that arrived on an inactive VP/VC pair. This address is looked up in the IP Routing Table (in hardware, as in [GuLK98]), thus specifying the desired outgoing VP; remember that choosing a VP corresponds to choosing a route in the ATM subnetwork (figure 2). Given a desired route (VPout), the next step is to find a VC identifier (ID) on that route for labeling the cells of this packet and differentiating them from those belonging to other packets, as explained in figure 1. This is done using the VC ID free list (figure 3), which may be a hardware-manipulated linked list or a bit mask and a priority encoder. If a free VCout is found, this VPout/VCout pair is written into the ATM connection table, at the entry corresponding to the VPin/VCin of the arriving cell, and the state of that entry is marked active. In this way, the current head cell of the IP packet effectively establishes a (temporary, wormhole) connection that will last until the tail cell of this IP packet is received. The head cell is then forwarded to the output, after undergoing the standard processing of active IP connections, i.e. VP/VC translation. Notice, however, that single-cell IP packets may exist, so the PTI field has to be checked, for head cells too; if PTI indicates an end-of-packet, then the newly established connection must be torn down right away (or it shouldn't be established to begin with). The Time-to-Live (TTL) field of the IP header is also contained in the head cell of the packet. If so desired, the routing filter can decrement this field (and drop the packet, as below, if the result is zero); this corresponds to each ATM subnetwork (figure 2) being counted as one hop in the path of the packet. Alternatively, if the entire ATM network is considered as a single hop, the TTL filed may be ignored.

If no free VCout exists for the desired VPout, the packet must be buffered until a VC ID is freed, when one of the other active packets that have been routed to the same VPout is fully received and forwarded. Alternatively, if such non-existence of a free VC ID is a rare event, we may simply drop the IP packet. The packet will also have to be dropped if the buffer is full (when there is no free VCout). In these cases, the state of this incoming connection must be marked as *buffer* or *drop;* in the case of buffering, a pointer to the proper buffer area must be inserted in the connection table, in lieu of VPout/VCout. The cost/performance tradeoff between buffering and dropping will be discussed in sections 3 and 4.

This concludes the general presentation on the operation of the wormhole IP over ATM routing filter. Next, we discuss some related issues: what happens with packets that arrive at the filter after one or more of their cells have been lost (dropped) in the ATM network; spreading of the packet's cells in time; quality of service and multicasting.

2.2 The Effects of Lost Cells

The discussion of section 2.1 and the flow diagram of figure 5 assumed that no cell of any IP packet is ever lost (dropped) in the ATM subnetworks. In reality, cells can be lost, so let us examine how the filter defined in figure 5 will perform in the presence of such losses. We can distinguish lost cells in three categories:

- *Middle cells:* cells other than head or tail cells of IP packets. Filter operation is unaffected by loss of such cells. The final destination will see an IP packet arriving with corrupted data content.
- *Head cells.* The filter of figure 5 interprets as "head" cell the first cell received on a particular VP/VC after a tail cell. When the real head cell of an IP packet has been lost, the filter will erroneously interpret another cell as "head", will extract an erroneous IP destination address from that cell, and will route the remaining cells of the packet in that direction. Eventually, a ("wrong") IP host will see an IP packet arriving with corrupted data content, while the real intended destination will never receive the packet.
- *Tail cells.* If a tail cell (a cell with its PTI field flagged as end_of_packet) is lost, the filter of figure 5 will interpret the cells of the next IP packet arriving over the same VP/VC as a continuation of the cells of that previous packet; in effect, the two packet (minus the tail of the first) will be concatenated into one packet. The final destination of the first packet will see a packet with corrupted data content; the destination of the second packet will never receive that packet.

In conclusion, the loss of one cell results in the corruption/loss of one IP packet –an unavoidable fact anyway– except that the loss of a *tail* cell results in the corruption/loss of *two* IP packets. When the cell loss probability, *CLP*, in the ATM subnetwork is low (e.g. 10^{-3} or less), and cell losses are uncorrelated to each other, the IP packet loss probability due to all of the above effects is approximately $(n+1) \cdot CLP$, where n is the number of ATM cells that the IP packet was segmented into. Out of this probability, the $n \cdot CLP$ portion is due to losses of this packet's cells, and the additional $1 \cdot CLP$ portion is due to the possible loss of the tail cell of a previous packet [HaSt98]. Thus, with an average packet size around 6 cells, tail cell losses account for an increase of about 16 % to the normal packet loss probability.

In connection with tail cell losses, we must also be careful concerning the following. If no next packet arrives over the VP/VCin of the lost cell for a long time, the outgoing VP/VC that was assigned to the previous packet is held in use (blocked) for a correspondingly long time. A time-out mechanism is thus needed to detect and release such long-held inactive outgoing VC's.

The above discussion concerned the wormhole IP filter as defined in figure 5. A number of enhancements are possible [HaSt98] in order to reduce the packet loss probability; given that this probability is already sufficiently low, however, these enhancements are not very appealing. First, the head cell of a segmented IP packet contains a packet length specifier. Based on this, the wormhole filter can compute the number of cells in the packet and initialize a counter in the connection table; each middle cell must decrement the counter. When the counter reaches zero, the incoming cell must be the end_of_packet; if the PTI field is not so marked, it means that the real tail cell was lost and this is the head cell of another packet. An artificial tail cell must then be generated for the previous packet (which is corrupted anyway), but the new cell can be handled intact, thus being saved from concatenation with the corrupted packet. Second, if LLC encapsulation is used before the the IP packet is segmented according to AAL5, then each head cell starts with the 8-byte sequence AA.AA.03.00.00.00.08.00 (hex). The routing filter can check for this sequence; if it not found, it means that the real head cell was lost, and this is in fact the second (or later) cell in the packet. In such a case, the entire (corrupted) packet can be dropped, thus saving the load of misrouted packets on the output network.

2.3 Spreading in Time of the Cells in a Packet

Do the ATM cells of each IP packet appear contiguous in time on the links of the ATM networks and wormhole routers, or are they spread in time as in figure 1? The degree of spreading, together with the sizes of the packets, affects the degree of interleaving among packets, hence the number of VC identifiers that are needed. We list here the factors that affect this spreading, and section 3.1 will come back on this issue. Figure 6 shows an "entry point" router, that is a filter whose input comes directly from a source or from a non-ATM network. Since all bits of an IP packet arrive here contiguously in time, it is natural for the ATM cells that result from its segmentation to also depart from the filter contiguously in time, in a burst, as in figure 6(a). Under these circumstances, a single VC per VPout suffices, since packets are never interleaved with each other in time like they were in figure 1. Under different circumstances, however, the cells of a packet may travel non-contiguously in time, and may well get interleaved with cells from other packets. One such reason is illustrated in figure 6(b): traffic shaping (policing) may be added to a routing filter, in order to avoid injecting bursts of cells into an ATM (sub-) network.

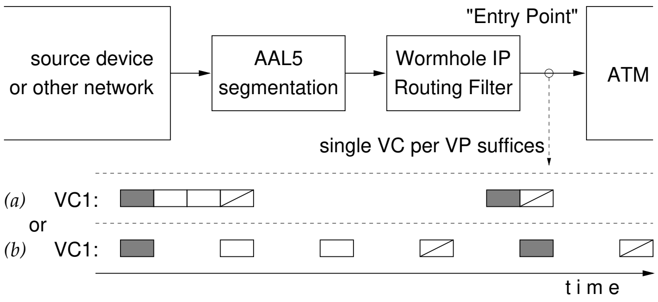

Figure 6: First entry point into ATM, (a) without and (b) with policing

Other factors leading to the spreading of the cells in a packet are shown in figure 7. Both of these have to do with the multiplexing of traffic from several sources onto a single link. In figure 7(a), cells from two different packets arrive (possibly contiguously in time) simultaneously from two different inputs, and become interleaved as they are all placed in a shared FIFO queue. In figure 7(b), an outgoing link is fed from multiple queues; these may correspond to different priority levels, or to per-flow queues with (weighted) round-robin scheduling. Cells from the different queues are multiplexed on the outgoing link, hence the cells belonging to a single packet will no longer be contiguous in time, even if they were so before the multiplexing. For example, high-priority native-ATM cells may interrupt a train of IP-generated cells. This is precisely one of the reasons for carrying IP over ATM: the network is shared among multiple types of traffic, and some classes of traffic (e.g. high-priority native-ATM) enjoy low-delay guarantees because their cells can interrupt long packets of the other traffic type (IP).

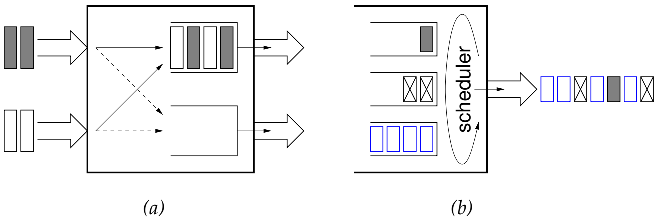

Figure 7: Factors leading to cell spreading

The inverse effect is also present: cells that have been spread apart in time can be brought back close to one another. As illustrated in figure 8(a), demultiplexing (departure of the dark cell in another direction) creates empty slots, while queueing behind other traffic squeezes out the empty slots and brings cells back next to each other.

Finally, at the "exit points" from the ATM network, IP packets must be reassembled from their ATM cells. A wormhole IP routing filter with buffer can accomplish that function too, as shown in figure 8(b). When a single VC identifier is available at the output of a routing filter, no

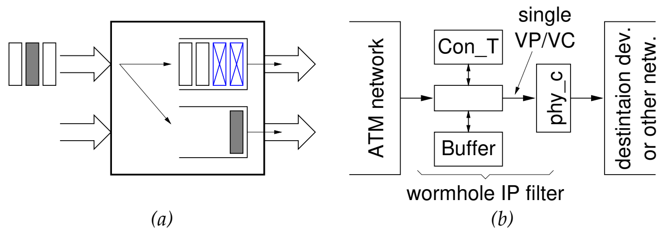

Figure 8: Cell consolidation in network (a), and packet reassembly at exit point (b)

packets can be interleaved on that output (or on that VPout), and hence packets are forced to depart sequentially. Of course, a wormhole IP filter that is configured in such a way will normally need to follow the buffering approach, rather than dropping cells when no available VCout is found (end of section 2.1). Notice that no IP routing table and no VC ID free list are needed in this filter configuration[3].

2.4 Quality of Service and Multicasting

The primary reasons for running IP on top of ATM is to get high throughput and quality of service (QoS) guarantees for traffic of various types in an integrated network. Wormhole IP over ATM provides an inexpensive method for fast routing of IP packets through ATM hardware and permanent VP's. In such an environment, QoS guarantees can be provided in a number of ways.

First and foremost, wormhole IP allows the coexistence of (segmented) IP traffic with native ATM connections. One approach, for example, is to use separate VP's for the IP packets and for the native-ATM cells; the former VP's can be managed as ABR or UBR connections (e.g. lower priority), while VP's carrying native-ATM traffic can be of the VBR class, for example (higher priority). This allows one unified network, where those who need QoS guarantees can get them by using ATM, while IP traffic gets what it always used to have: a best effort service with no guarantees.

Rather than providing native-ATM traffic with an absolutely higher priority relative to IP packets, one may opt for (weighted) round-robin scheduling between the two kinds of VP's, or for plain FIFO scheduling without consideration for VP type. In such an environment, in order to be able to offer QoS guarantees to native-ATM connections, all traffic (including IP) must be policed. It is possible to perform such policing for the IP-carrying VP's in the wormhole routing filters; this will be discussed, along with simulation results, in section 3.4.

[3] in some respect, this configuration is the complement of the one at "entry points" (figure 6); at entry points, no connection table and no buffer memory are needed.

To differentiate the IP packets among themselves, in order to preferentially provide QoS to some IP flows, a method for IP flow classification is needed. Such classification capability can be added to the wormhole routing filters. In that case, the outgoing VP for an IP packet is selected based on both the routing table outcome and the flow classification outcome: multiple VP's are provided for each destination, each VP corresponding to a different service class in the ATM subnetworks. In fact, it suffices for flow classification to be performed at "entry points" only (figure 6), leading to the selection of the appropriate *class of VP's* for each IP packet. Downstream wormhole routing filters will choose the outgoing VP class based on the incoming VP class, and then the routing table will choose a specific VPout within the given VP class.

Multicasting becomes increasing important in modern networks. Efficient multicasting requires "branching points" (switches) in the network to have the capability to copy traffic onto multiple outgoing links; multicasting by generating multiple copies at a source and then forwarding all of them on the same link is inefficient. In an environment such as the one of figure 2, all branching points are in the ATM subnetworks. Thus, efficient multicasting can be supported using the ATM switches; supporting multicasting in the wormhole routing filters would be inefficient, since it would involve generating multiple copies of cells and forwarding them over the same (single) outgoing link. Multicasting for IP packets can be implemented by setting up special multicast VP's in the ATM subnetworks, and then inserting these into the IP routing tables of the wormhole filters.

3. Performance Simulation

How many VC identifiers are needed in each outgoing VP for a wormhole IP over ATM routing filter to perform well? In a bufferless filter, packets are dropped whenever free VC ID's are exhausted, so we would like to have enough VC labels for this to be a rare event. In a routing filter that can buffer cells in expectation of a VC ID becoming available, cells are delayed and buffer occupancy grows when VC ID's are exhausted, so again we would like this not to happen too often. In the extreme case of a single outgoing VC (e.g. at an exit point –figure 8), cells are effectively reassembled into packets, in the buffer memory, before being allowed to go out. What is the packet delay, and how much buffer space is needed in these latter two cases?

To answer these questions, we simulated the operation of a wormhole routing filter like the one defined in section 2, under a traffic load that we generated based on real IP traffic. We simulated the routing filter operation over a time interval of 300,000 cell times. The input trace was generated as presented in section 3.1, based on real IP traffic at the FIXWEST West Coast federal interexchange node in January 1997 [NLANR97]. We preferred FIXWEST traces because they were taken from IP backbone routers, and are thus more representative of the overall IP traffic. Out of the information provided for each packet in the trace, we only needed and used the arrival timestamp, the IP destination address, and the packet length. The first few seconds of the trace sufficed for us to fill our simulated time of 300,000 cell times with IP traffic.

3.1 Traffic Characterization and Input Trace Generation

The larger an IP packet is, the longer its cells will hold the VC label assigned to them, thus making this label unavailable for other packets. In the trace used (first 8 seconds), the average IP packet size, measured in terms of the ATM cells into which it is segmented, is 6.1 cells. Thus, *packets are small,* which is *good news* for wormhole IP over ATM, because it means that VC identifiers are not held for long periods by the same packet. To get an idea about the variance of the packet size, figure 9 plots the size of the largest packet seen to arrive in each 5 ms time window, for the 1600 such windows in the first 8 seconds of the trace. Obviously, there are many very small packets, given that the average packet size is about 6 cells. On the other hand, packets larger than 32 cells are extremely rare, while packets in the range 14 to 32 cells occur in several but not in the majority of the 5ms windows[4].

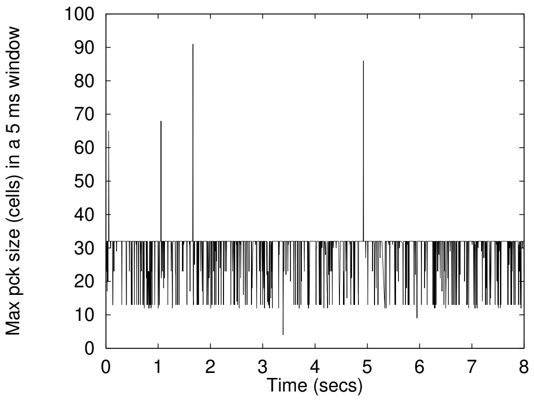

Figure 9: IP packet size

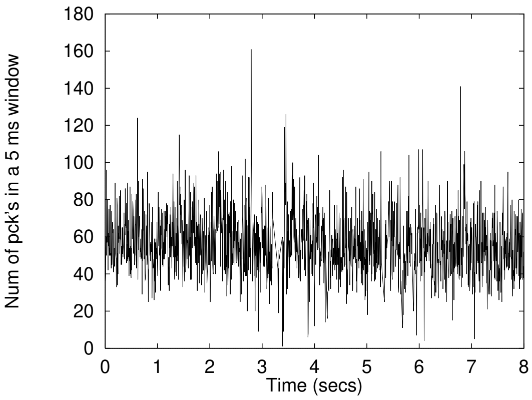

Figure 10: Traffic burstiness

The larger the number of packets transmitted in a given time period, the more the VC identifiers that we are likely to need for them. The average number of packets per unit of time, obviously, depends on the throughput of the network link considered and its average load. To get an idea

[4] the largest ethernet packet (1500 bytes) is segmented into 32 ATM cells; 576-byte packets (rfc894 "off ethernet" MTU) yield 13 ATM cells each; 552-byte packets (512 bytes of user-data) yield 12 ATM cells each; see http://www.nlanr.net/NA/Learn/Gm/pktsizes.html

about the variance of the load, figure 10 plots the number of packet arrivals in each 5 ms time window for the 1600 such windows in the above trace (the window size is narrow enough to account for just a few tens of packet arrivals). We see that busy periods are about 3 times busier than average, and light-load periods are about 3 times less busy than average, i.e. a load fluctuation roughly in a range of 10:1.

Going from the original IP trace to a realistic pattern of cell arrivals for the wormhole routing filter under consideration is not easy. We assume that all the IP packets in the trace pass through the routing filter. We calculated the arrival time of the head cell of each packet from the arrival timestamp of that packet in the trace, by performing a linear scaling of the time axis. Four different scaling factors were chosen, in such a way as to yield average loads of 25%, 50%, 75%, and 85%, due to IP traffic, on the incoming link, over the 300,000 cell-time duration of the simulation input that we generated. On top of this IP load, we add a native-ATM traffic load of 2/3 of the remaining link capacity, i.e. 50%, 33%, 17%, and 10%, respectively. Thus, the total link load comes up to 75%, 83%, 92%, and 95%, in the four loading scenarios that we used. These are shown in figure 11.

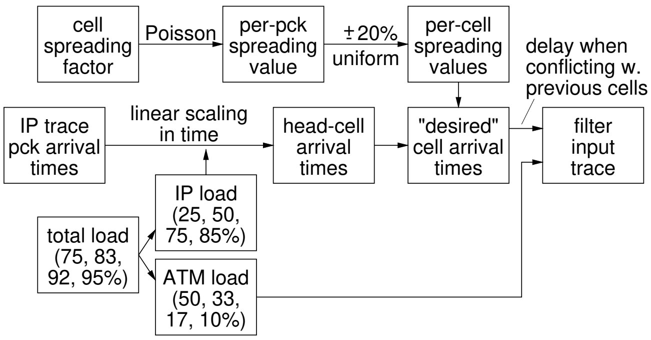

Figure 11: Input trace generation steps

Having specified the arrival time of each head cell, as above, we now need to find the times at which the other cells of the packets arrive at the routing filter. This, obviously, depends on the ATM subnetworks through which the IP traffic has passed, their topology, their load, their policing and scheduling schemes, as discussed in section 2.3. It would be very difficult for us to simulate such an ATM subnetwork, and the conditions of such a simulation would be questionable anyway. Instead, we use a parameter, which we call cell *spreading* factor, and, based on this, we generate an input trace, as outlined in figure 11 and described below. For each IP packet, we pick a random spreading value according to a Poisson distribution with average value equal to the spreading factor. To find the arrival times of the cells of a packet, we start by spacing these cells in time at a distance between each other equal to the spreading value chosen for the packet; then, we "move" each cell, in time, by a random number uniformly distributed between -20% and +20% of the spreading value of the packet. The rationale is the following: different packets have traveled through potentially different routes, so they each have a different spreading value; all cells of one packet, though, have traveled through a common route, in close time proximity, and have thus encountered similar traffic conditions, hence their inter-cell spreading is correlated. The final step is to find a specific cell-time slot where to place each cell of each packet, in the generated input traffic pattern. According to the head-cell arrival times specified by the scaled trace, and according to the randomized spreading process, the arrival times of cells from multiple packets may collide with each other, and may also collide with the arrival times of native-ATM cells. In such collision cases, the cell is placed in the next empty slot, after the native-ATM cells[5] and the cells of previous IP packets have first been assigned to their slots in the time axis.

The most interesting number that characterizes the resulting input trace is the degree of *interleaving:* at each point in time, the degree of interleaving is the number of "active" packets, that is, packets for which the head cell was already received while the tail cell has not yet arrived. This coincides with the number of incoming IP-type VC's that are in the active or drop/buffer states (figure 4). The degree of interleaving is a useful figure because it gives an indication on the number of VC's required at the output. In fact, if the number of VC identifiers (for IP traffic) for each outgoing VP exceeds the maximum degree of interleaving, then *all* cells are guaranteed to depart immediately without ever needing to drop or buffer any cell. Of course, usually, quite less VC's per VPout suffice, since the active packets are distributed over multiple outgoing VP's.

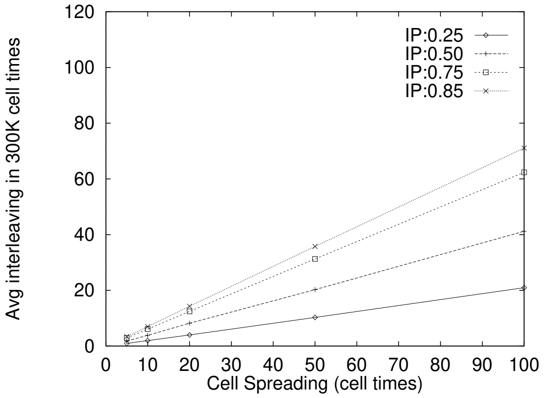

Figure 12: Average packet interleaving in incoming stream

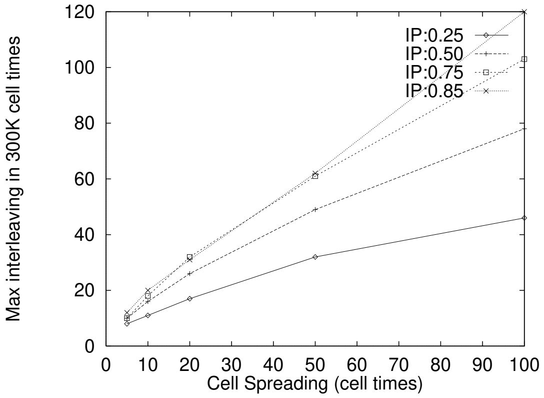

Figure 13: Maximum packet interleaving

[5] the current simulation results are from runs in which IP cells were placed before native-ATM cells. We are now in the process of making new simulation runs, with the placement order as listed in the text.

Figure 12 plots the average degree of interleaving and figure 13 gives the maximum degree of interleaving over the 300,000 cell times of the generated input trace, as a function of the spreading factor, for the four different loads considered. As expected, interleaving is roughly proportional to the load and to the spreading factor. The values of spreading considered in our simulations is quite realistic, we believe. In lightly loaded ATM networks, factors like those considered in figure 7 should give spreading values in the range of 2 to 10. Under heavier load, or with per-VP policing, we expect higher spreading factors. A spreading of 100 is rather extreme: it corresponds to hundred-fold multiplexing (figure 7) without any intervening consolidation (figure 8(a)). Of course, for packets originating from very slow networks, the spreading factor will be large, so this must be taken into consideration in such environments; wormhole routing filters can be configured so as to reduce the spreading factor at their output, by providing buffer space and restricting the number of outgoing VC's per VP (section 3.3).

Another difficult decision in our simulations was the *routing function* to be used (the contents of the routing table). In a real situation, this is highly dependent on the topology and location in the internet. We decided to use a routing function that selects the same route for hosts with nearby IP addresses, while the routes for hosts with distant IP addresses are (pseudo-) uncorrelated. The routing function used is as follows: for a configuration with n outgoing VP's, route a packet destined to IP address d through VP number (d **div** 57) **mod** n, where **div** is the integer quotient of the division, and **mod** is the integer remainder of the division; in other words, every 57 neighboring IP addresses are assumed to be in the same direction. The number 57 was chosen experimentally: it yields almost equal average load on all outgoing VP's.

The number of outgoing VP's is in general expected to be relatively high: it is approximately the number of ports of an ATM subnetwork. We want these subnetwork to be large, so that fewer routing filters are needed; on the other hand, the subnetworks cannot be too large, because then we run out of permanent VP's when connecting port pairs. We expect numbers between several tens and a few hundreds of outgoing VP's. Simulating a large number of VP's was hard, because the traffic on each of them became so sparse that our results were questionable. Thus, we restricted ourselves to at most 32 VPout; it is easy to extrapolate from our results to larger numbers of outgoing VP's.

3.2 Bufferless Routing Filter Performance

Given the short average length of IP packets and the resulting relatively low degree of interleaving, it becomes attractive to make *bufferless* wormhole routing filters, since that reduces the hardware cost (section 4.1). How many VC's per outgoing VP are needed for such a bufferless filter to rarely have to drop packets due to unavailability of VC identifiers? Figures 14 and 15 plot the packet loss probability (PLP). We see that 16 VC's per outgoing VP are enough for the PLP to never exceed 0.1% for spreading factors up to 100. For lower loads and lower spreading factors, even less VC's suffice. For fewer outgoing VP's, figure 15 indicates that more VC's per VP are needed to achieve comparably low PLP, but the total number of VC's in all outgoing VP's can be lower. By extrapolation, we conclude that 8 to 16 VC's per VP suffice when the number of VP's is in the hundreds; when spreading factors in excess of 100 are expected, one should consider having 16 to 32 VC's per VP.

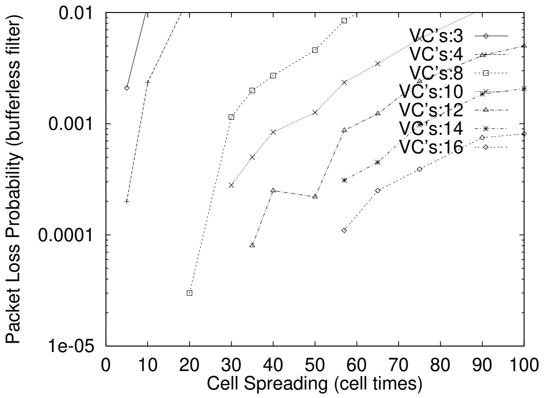

Figure 14: PLP for 75% IP load and 32 VP's

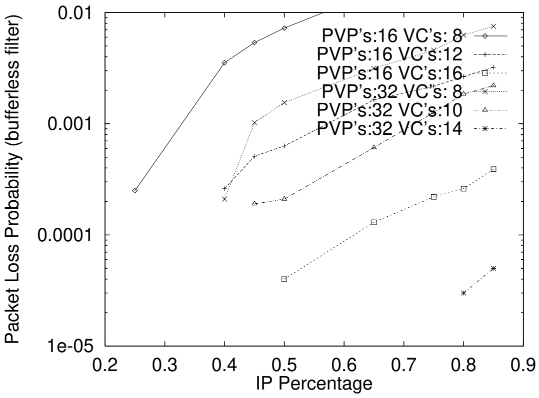

Figure 15: PLP versus IP load for spreading=50

In conclusion, a total number of VC identifiers for IP traffic on the order of a few thousand suffice for a bufferless routing filter to perform very well, with up to a few hundred outgoing VP's. These numbers are very realistic, and easily achievable with low-cost hardware (section 4.1). One would worry if packets dropped were preferentially long packets; if this were true, long packets would have a hard time passing through the filter, even after retransmission. However, this is not true: packets are dropped when all VC's have been exhausted by the packets *ahead* of themselves; thus, the decision to drop or not to drop depends entirely on the previous packets, and not at all on any property of the packet to be dropped.

3.3 Performance of Filters with Buffer

When the VC labels in the desired outgoing VP have been exhausted, the alternative to dropping the packet is to buffer its cells until a VC identifier becomes available. The buffer management algorithm that we studied and simulated is as follows. For each VPout, a queue of buffered packets is maintained. For each buffered packet, a queue of cells is maintained. When the head cell of a new packet arrives, it is immediately forwarded to the output only if a free VCout exists *and* no other packets destined to the same VPout are currently buffered; this latter condition (which was not shown in figure 5, for simplicity) is needed to guarantee order preservation and non-starvation of buffered packets. Otherwise, the new packet is buffered at the tail of the queue for its VPout. When cells of buffered packets arrive, they get buffered at the tail of the queue for their packet (the connection table points to that queue). When a VC label for a certain VPout is freed, the front packet in its queue of buffered packets is examined; if no VC label has yet been assigned to that packet, then this label is given to it, and the VP is marked as *ready* for transmission of buffered cells (see below).

Output scheduling, in each cell time, is as follows. If there is an incoming cell of native-ATM type, that cell is forwarded to the output. If there is an incoming cell of IP type and active state, that cell is forwarded to the output. Otherwise (i.e. when there is no incoming cell, or the incoming cell must be buffered), one of the previously buffered cells is forwarded; that cell is chosen as follows. A circular list of VP's that have buffered packet(s) and are *ready,* for transmission is maintained; a VP with buffered packet(s) is ready for transmission when a VC label has been assigned to the front packet in its queue of buffered packets. The buffered cell to be forwarded to the output is the front cell in the queue of the front packet of the next VPout in the circular list of ready VP's; after forwarding this cell, the pointer of the circular list is advanced to the next ready VP[6]. In this way, the filter interleaves buffered cells from all ready VP's, in a round-robin fashion; this is done in order to avoid bursts of cells of the same VP, i.e. in order to smooth the traffic injected into the ATM subnetwork. Within each ready VP, all buffered cells from *one* buffered packet (the front packet in the queue) are forwarded, before forwarding any buffered cell of any other buffered packet (even if more VC labels have become available in the meanwhile). This is done in order to reduce the degree of interleaving and the spreading factor on this outgoing VP. Notice that once there are buffered packets for a given VPout, all newly arriving packets destined for that VPout get buffered, and will then start departing in strict arrival order. When the last buffered cell of a buffered packet is forwarded to the output, if that was the tail cell of the packet, the packet is dequeued and its VC label is given to the next buffered packet for that VP. If this was not the tail cell of the packet, the packet is dequeued but its state is marked active in the connection table, so that further cells of this packet will cut-through to the output; the VC label is held by this packet; if another free VC label exists for that VPout, that label is given to the next buffered packet for this VPout.

[6] the current simulation results are from runs that followed a different scheduling algorithm: the pointer of the circular list is advanced to the next ready VP only after all cells of this packet have been forwarded. We are now in the process of making new simulation runs for the scheduling algorithm described in the text.

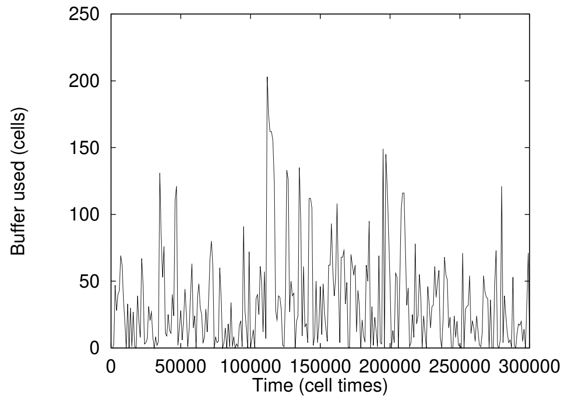

Figure 16: 32 VPs × 2 VCs each; 75% IP load; spreading=20

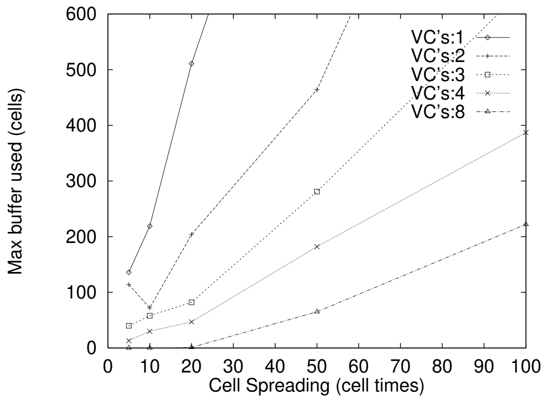

Figure 17: 32 VPs, 75% IP load

For a routing filter that buffers cells in the absence of outgoing VC labels, the first question concerns the amount of buffer space needed. Figure 16 plots buffer occupancy as a function of time for some representative traffic parameters but with just 2 VC's per VP. Even with so few VC identifiers available, a buffer space of just 200 cells (11 KByte) suffices. Figure 17 gives the maximum buffer occupancy for various spreading factors and number of VC's per VPout. Figure 18 does the same for various IP traffic loads. We see that a few tens of Kilobytes of buffer space suffice even when there are as few as 4 VC labels per outgoing VP.

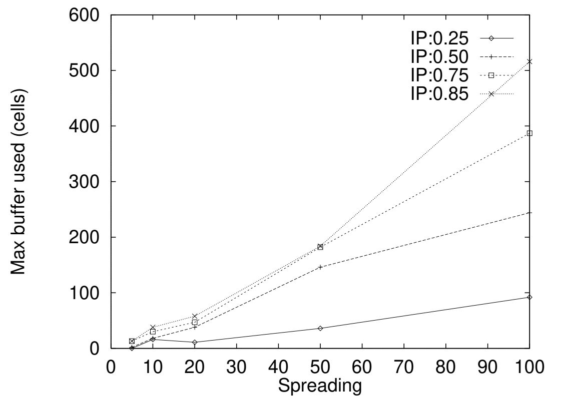

Figure 18: 32 VPs × 4 VCs each

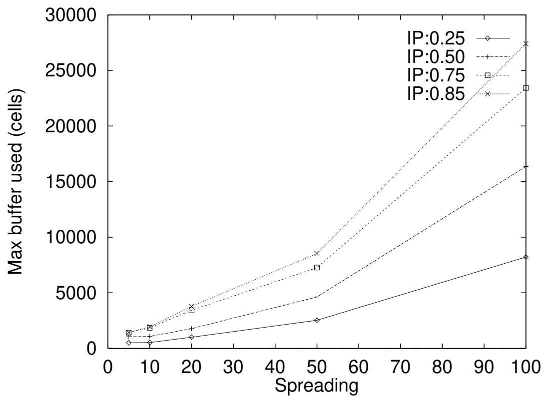

Figure 19: Reassembly buffer space: 1 VPout, 1 VCout

A special case of interest is IP packet *reassembly* at an *exit point* from the ATM network (figure 8(b)). The routing filter will automatically perform packet reassembly when configured with a single outgoing VP/VC pair. Figure 19 plots the buffer space needed in this case for various spreading factors and IP traffic loads. We see that quite larger buffers are needed now: several thousands of cells (hundreds of Kilobytes) are needed for spreading factors up to 50, while spreadings of 100 require one or a few Megabytes of buffer space for packet reassembly.

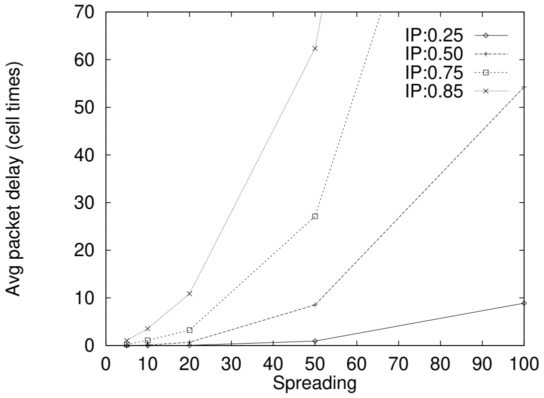

Figure 20: Average delay (32 VPs × 4 VCs each)

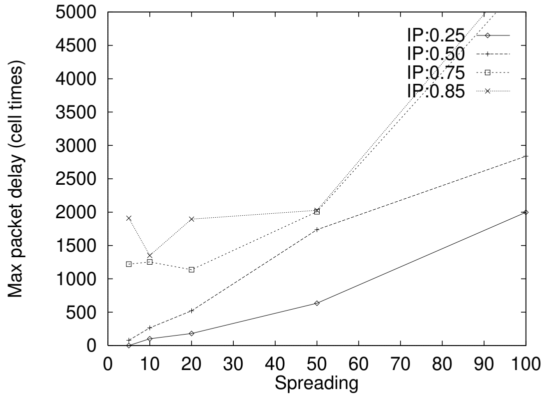

Figure 21: Worst case delay (32 VP × 4 VC)

The other performance metric of interest, for a routing filter that can buffer cells, is the delay caused by such buffering. Considering that a packet transmission is complete when its tail cell has been transmitted, we measured the delay of each packet as the time elapsed between its tail cell arrival at the filter and the departure time of that tail cell from the filter. Figure 20 plots the average packet delay, and figure 21 plots the maximum (worst case) delay seen during the 300,000 simulated cell times. We observe that the worst-case delay is about *two orders of magnitude* worse than the average packet delay. This is important: it tells us that, when there are insufficient VC labels, most packets suffer very little, but some packet suffer a lot, thus making QoS quite unpredictable.

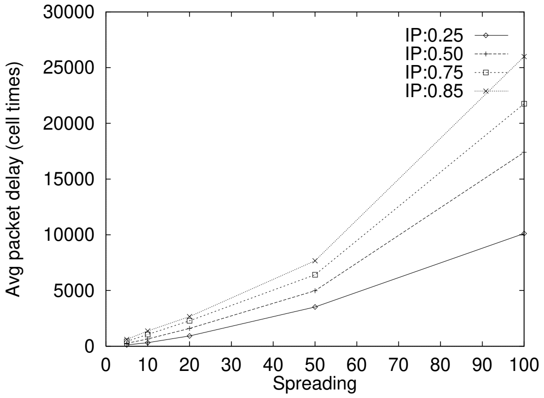

Figure 22: Average reassembly delay (1 VPout, 1 VCout)

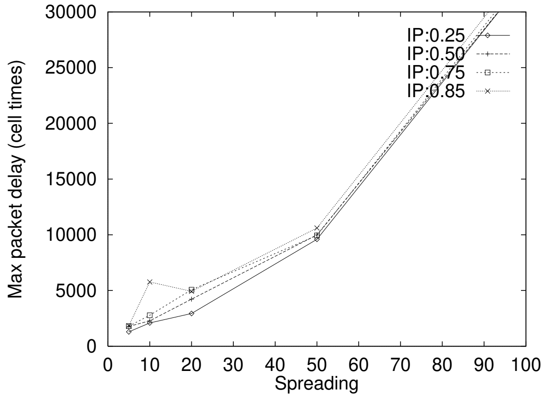

Figure 23: Maximum reassembly delay

Finally, figures 22 and 23 plot the delay incurred by packet reassembly, at an exit point with 1 VP and 1 VC. We see that in the case of reassembly both the average and the maximum delay are large; the worst-case delay is only about twice longer than the average. The delays encountered are in the thousands or tens of thousands of cell times (milliseconds, for 622 Mbps links); only for very small spreading values do we see delays in the 0.1 to 0.5 ms range. It is important to realize that *wormhole IP over ATM saves us* from precisely these *reassembly delays* at every IP router encountered on the path of a packet!

3.4 Policing, per Outgoing VP

For reasons of quality of service in the ATM network, one may want to perform *traffic shaping (policing)* for the VP's that carry IP traffic too. Wormhole routing filters are convenient places to perform this. We simulated a *leaky bucket stage,* placed right at the exit of the routing filter considered and simulated above. In this simulation, there is a leaky-bucket controller for each outgoing VP. Every cell that is allowed to depart from the basic routing filter is then subject to policing: if it finds a credit in the leaky bucket of its VP, it takes that credit and is allowed to go out; otherwise, it has to wait (in a per-VP queue) for such a credit. We simulated bucket sizes (burstiness factors) of 1 or 5. The rate of credit generation (bucket fill rate) was set at:

$$\textit{Rate Limit per VP} \;=\; \frac{\textit{Average IP Traffic Load}}{\textit{Number of Outgoing VPs}} \times \textit{Rate Factor}$$

If the cells were uniformly distributed over all outgoing VP's and over the entire simulation interval, then a *Rate Factor* of 1 would just suffice for the cells to go through the leaky bucket controller with negligible delay. Of course, no such uniform distribution exists, neither in VP's nor in time, so rate factors significantly greater than 1 are needed in order to avoid excessive delays due to policing.

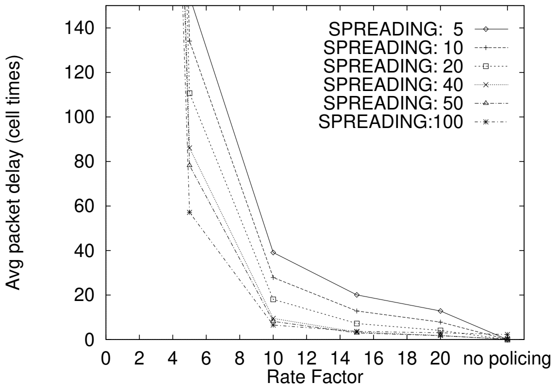

Figure 24: Leaky bucket with burst=1 (50% IP load, 32 VPs × 8 VCs)

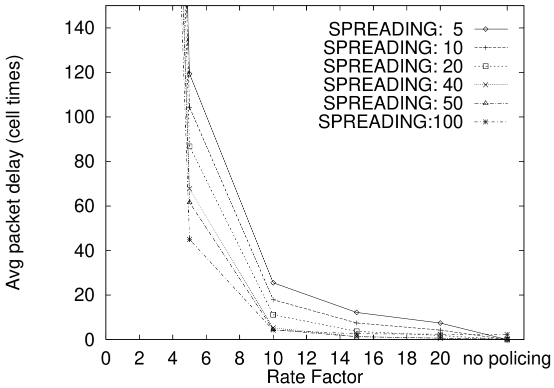

Figure 25: Burst=5

Figure 24 shows the average packet delay, with policing and various rate factors, and without policing; the burst size is 1 (no burstiness allowed). Figure 25 shows the same numbers when bursts of 5 cells are allowed by the leaky buckets. A rate factor of 10 corresponds to a rate of

$(50\% \ / \ 32) \times 10 \approx 16\%$ for each (per-VP) leaky bucket. For such a policing rate, we see that the average packet delay does not exceed 40 cell times, and goes down to 25 when a burstiness of 5 is allowed. For spreading values of 40 or more (incoming traffic), the average packet delay is less than 10 cell times (larger spreadings roughly correspond to stricter policing upstream in the network). For this same rate factor of 10, the maximum packet delay seen during the simulation (not shown here) ranged from 500 (large spreadings) to 1000 (small spreadings). For a rate factor of 5, corresponding to a limit of 8% of the outgoing link capacity per VP, the average packet delay was on the order of 100 cell times, while the worst case delay ranged around 4,000 cell times. The total buffer space needed (not shown in the figures), for the basic routing filter as well as for the policing queues, is less than 200 cells for a rate factor of 10, and grows to 400 cells for a rate factor of 5, and further to 1,000 cells when the rate factor drops to 3.

4. Hardware Complexity Analysis

To estimate the cost of wormhole IP over ATM, we conducted an initial hardware design study. We start with a 622 Mbit/s bufferless filter, since that is simpler and performs quite well (section 3.2); then we consider routing filters with buffer and pipelined operation at rates up to 10 Gb/s.

4.1 FPGA-based Bufferless Routing Filter at 622 Mb/s

Figure 26 presents a block diagram of a bufferless routing filter. The largest memory needed is the first stage of the Routing Table, with a size of 16 Mwords by 16 bits each (10 bits for VPout, 1 validity bit, 5 depth bits [GuLK98]); it can be implemented using 4 DRAM chips of size $16M \times 4$ each. Next, we need the second stage of the Routing Table and the Connection Table. These could have a modest size and be placed in SRAM, but we opted for a more comfortable allocation in DRAM; this allows us to provide a very large connection table –much larger than needed for wormhole IP purposes– so that native-ATM traffic can take advantage of it. The connection table can be organized, for example, as 1M entries by 31 bits each: 3 bits for the connection type and state, plus 28 bits of fully translated VP/VC. For native ATM cells, we need to read two consecutive words; for wormhole IP cells, the first word suffices (3 bits of type/state, 13 bits of outgoing VP/VC). For the second stage of the Routing Table, 2M entries (10 bits for VPout and 1 validity bit each) are plenty enough. This latter memory and the Connection Table are placed inside a single $4M \times 16$ DRAM chip. To economize on the number of pins, all DRAM chips share address and data buses; this reduces parallelism, but satisfactory performance can still be achieved, as will be seen shortly.

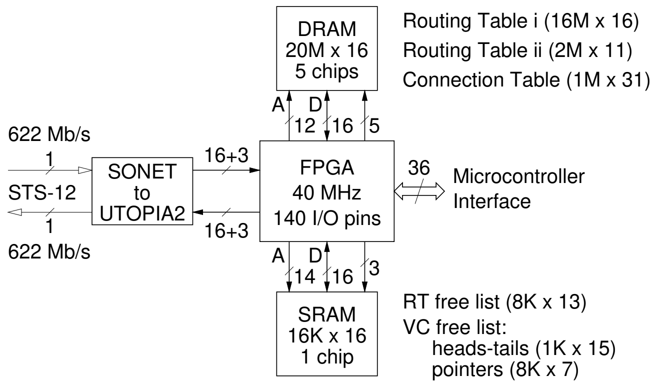

Figure 26: Block diagram of 622 Mbps bufferless wormhole routing filter

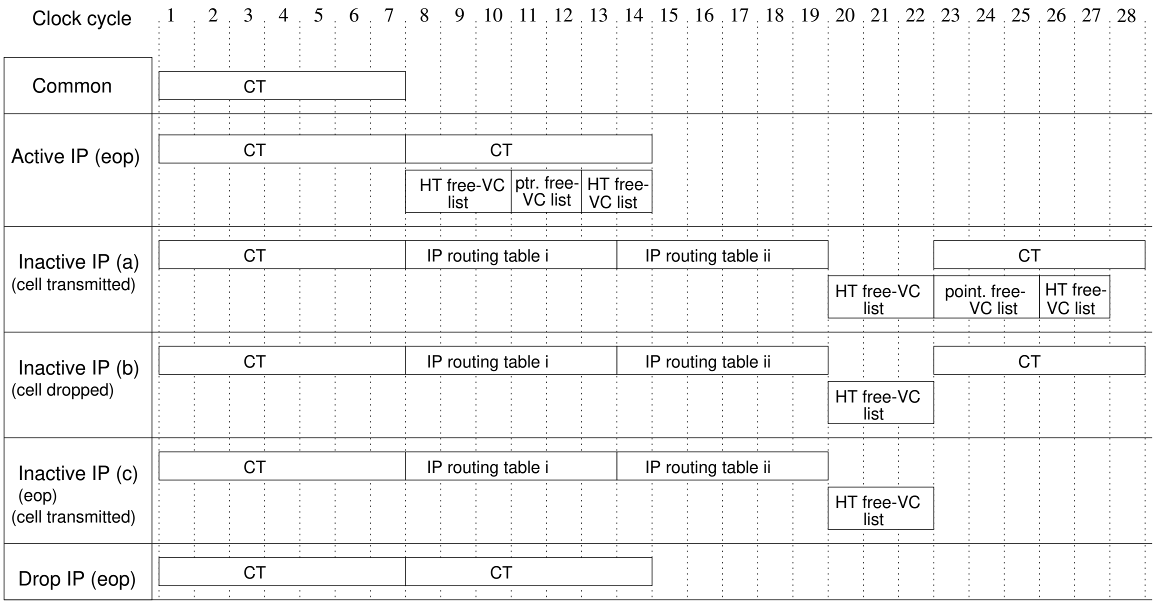

Figure 27: Memory access timing for the bufferless wormhole routing filter

In this example design we allow up to 1024 outgoing VP's, and up to 8K VC labels for IP traffic in all of the outgoing VP's. Thus, one can have configurations with 1K outgoing VP's $\times$ 8 VC's per VP, or 512 VP's $\times$ 16 VC's, or 256 VP's $\times$ 32 VC's, and so on; as we saw in section 3.2, these yield excellent performance. With these sizes in mind, the connection table contains 13 bits of outgoing VP/VC (for IP connections), and we need up to 1024 free lists of VC identifiers (one per VP), and up to 8K free VC ID entries. The latter two are included in an SRAM chip (figure 26), together with the free list for the second stage of the routing table [GuLK98]. Cell processing, header translation, and forwarding are performed by a field programmable gate array (FPGA) chip. The FPGA connects to the DRAM and SRAM chips, and to a SONET/UTOPIA-2 conversion chip; a microcontroller may also have to be included in the routing filter, if it turns out to be too complicated for the FPGA to maintain the Routing and Connection Tables by itself, according to instructions that it receives through the ATM link. These connections amount to using, e.g., a 208-pin FPGA (like the Altera EPF10K40-RC208), with at least 140 user I/O pins.

Timewise, we need 28 clock cycles for the worst case operation (open new IP connection) that has to fit inside a single cell-time. The corresponding timing diagrams are given in figure 27. Since the cell time is approximately 700 ns for 622 Mb/s, the clock period can be as long as 25 ns, requiring a clock frequency of 40 MHz or higher. We have assumed that a DRAM access takes 6 clock cycles (150 ns); reading two consecutive words from the connection table is assumed to take 7 cycles. SRAM access can, in principle, be performed in 1 clock cycle, but we conservatively allowed 3 cycles per SRAM read and 2 cycles per SRAM write, in order to account for the long FPGA setup and output delays and the consequent need to use pipeline registers at the FPGA-SRAM interfaces.

4.2 *Hardware Cost of Routing Filter with Buffer*

Figure 28 shows the block diagram of the routing filter that results when we add cell buffering capability to the filter of figure 26. The DRAM memory is the same in the two filters. In figure 28 we have added a buffer memory SRAM. To provide enough throughput for the incoming and outgoing traffic, it must have a width of 32 bits; we used two 32K × 16 chips, thus providing buffer space for 2K cells, which is more than adequate (section 3.3).

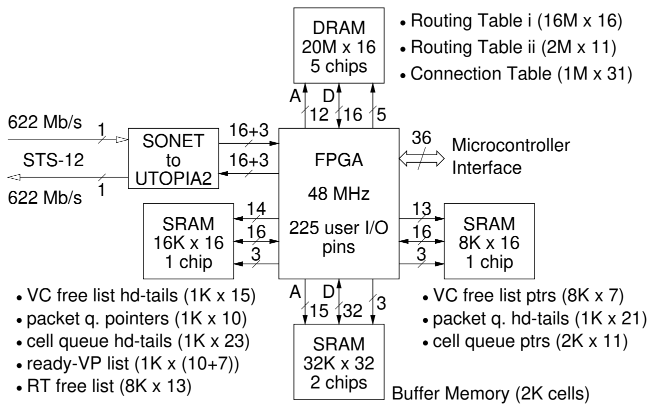

Figure 28: Block diagram of 622 Mbps wormhole routing filter with buffer

The data structures that are necessary in order to manage the buffered cells, along with the RT free list and VC free list that were also present in figure 26, are maintained in two additional SRAM chips; these need to be separate in order to provide enough parallelism. The additional data structures, in figure 28, are: *(i)* the packet queues: there can be at most 1024 of them, one per

VPout, and at most 1024 buffered IP packets can be kept track of in them; *(ii)* the cell queues: there can be at most 1024 of them, one per buffered IP packet, and they keep track of the buffered cells (up to 2K); and *(iii)* the circular list of ready VP's. Because of the 3 additional SRAM chips, the FPGA now needs 225 user I/O pins versus only 140 I/O pins in figure 26.

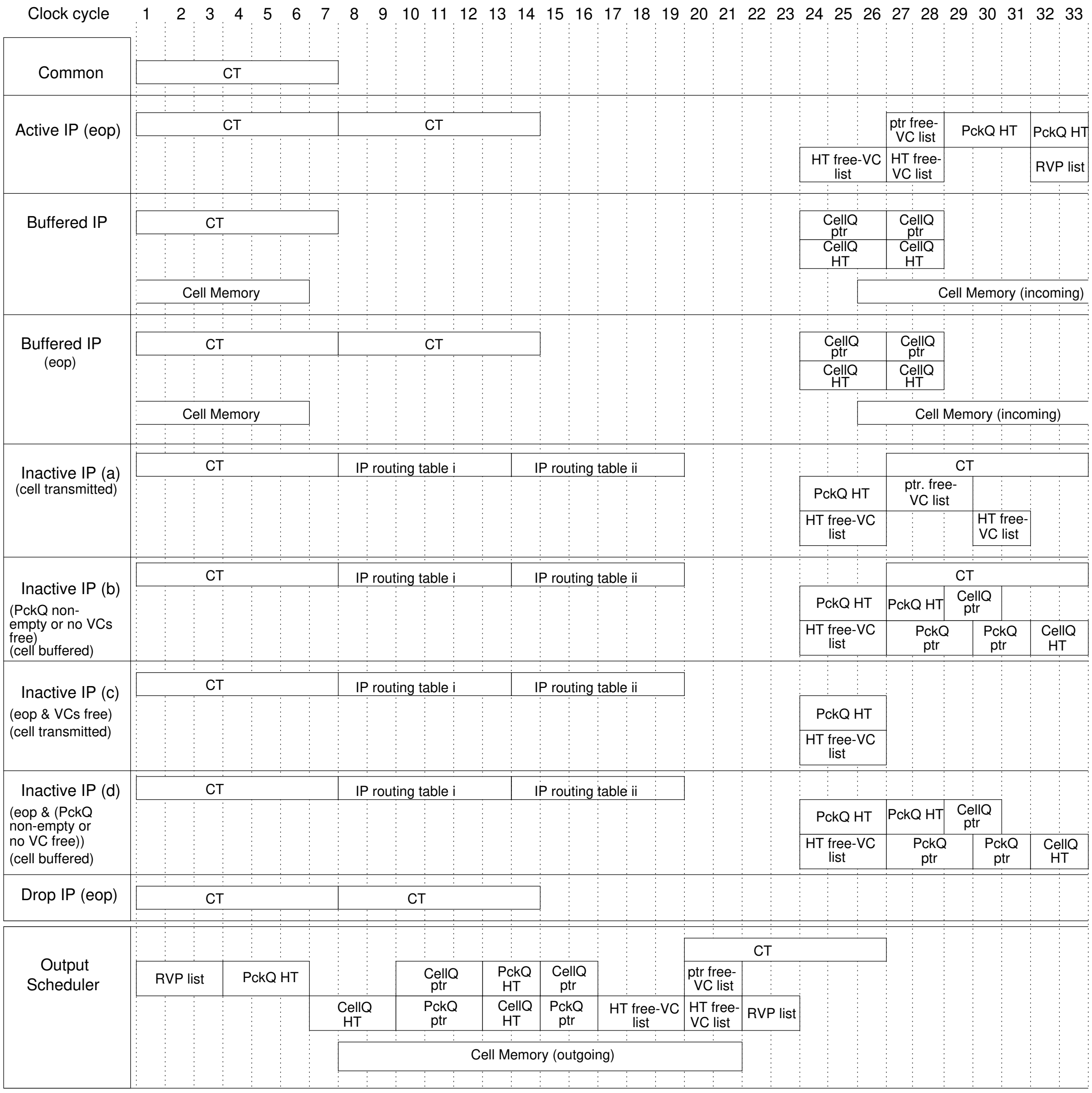

Figure 29: Memory access timing for the wormhole routing filter with buffer

Figure 29 shows the timing diagrams for the memory accesses during the various operations. We now need 33 clock cycles per cell-time, leading to a requirement for 21 ns clock period, or 48 MHz clock frequency; upper-end, modern FPGA's can reach these requirements. As in section 4.1, we assumed 6 clock cycles (only 125 ns, now) per DRAM access, 3 cycles per SRAM read, and 2 cycles per SRAM write (however, accesses to the buffer memory are sequential, and are thus assumed to proceed at the rate of 1 per clock cycle). If an ASIC is used in place of the FPGA, some or all of the SRAM's can be placed on-chip, thus saving on pins and clock cycles.

Overall, we see that a routing filter with buffer is appreciably more complex than a bufferless one. In terms of component cost, we need an FPGA with 60% more pins; we also need 30% more memory chips. Although these may not be too significant, the considerably higher design complexity of a filter with buffer must also be taken into consideration. In conclusion, we see no serious reason for going to a filter with buffer, unless one wants to support packet reassembly at "exit points", in which case large buffer space is needed (section 3.3).

4.3 Pipelined Operation at 2.5 and 10 Gbit/s

The routing filters considered in sections 4.1 and 4.2 used very little parallelism; that is adequate at 622 Mb/s, and simplifies the design. On the other hand, the wormhole routing filter operation is simple enough for it to be implementable, even today, at much higher speeds as well, by using more parallelism. Given the relatively small sizes of the connection table and VC free list, according to the numbers seen in section 3, these memories can be easily implemented on-chip, at high speed, and pose no serious problem for operation at multi-gigabit-per-second rates. The hardest problem comes from the IP Routing Table (RT). Fortunately, the approach of [GuLK98] is scalable to very high speeds. At 2.5 Gbit/s, the cell time is around 170 ns. Two DRAM accesses, as required for the RT, may fit in 170 ns; if not, the second stage of the RT could be implemented in SRAM, or the two parts of the RT can form two stages of a pipeline. At 10 Gb/s, the cell time is 42 ns, approximately. At this speed, each stage of the RT can be copied into e.g. two separate DRAM's, each with a cycle time of 84 ns; each DRAM is accessed in alternating cell times.

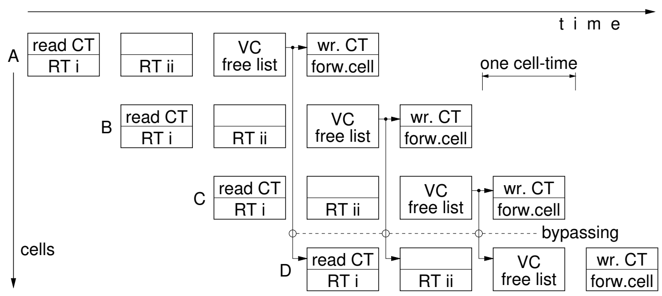

Figure 30: Example of pipelined operation of a bufferless wormhole routing filter

Once the IP Routing Table (RT) is made to have enough throughput, using techniques such as the above, the operations of the routing filter can be arranged in a pipeline. Figure 30 shows an example of such a pipeline, where each stage corresponds to one cell time (170 or 42 ns). In this example, RT accesses take up to two stages. The Routing Table is read in parallel with the Connection Table (CT): we always and "blindly" start a routing look-up before knowing if the current cell is a head cell; if it is not, just throw away the result of the look-up. In the third stage, if the CT indicated a head cell, we get a free VC in the VPout chosen by the RT. The cell is forwarded out in the fourth pipeline stage; if this was a head cell, the CT is updated with the new VP/VC pair. As in most pipelines, there are dependencies among the multiple cells that are being processed in parallel, in the various pipe stages; these dependencies, however, can be resolved using bypassing −a technique commonly used in all processor designs today. As shown in figure 30, the CT entry for cell D is read before cells A, B, and C have had the chance to update the CT −something they should normally have been allowed to do. If cell D belongs to the same VP/VCin as cell C, or B, or A, and if that earlier cell was a head cell that selects a new VP/VCout in its third stage, then cell D must use that VP/VCout; this information is readily available in the latter stages of the pipeline, and can be bypassed (forwarded) to the earlier stage where cell D is being processed, as indicated by the arrows in figure 30. Microprocessors with pipeline cycles under 10 nanoseconds are routinely designed today; it follows that wormhole IP over ATM operation at OC-192 rates (10 Gigabits per second) poses no serious problem with its 42 ns pipeline stage.

5 Related Work and Conclusions

A conventional IP router is a device that receives IP packets, looks up the destination address in a routing table in order to choose the proper outgoing link for each packet, and forwards the packets on these links; when output contention occurs, it temporarily buffers packets. A combination of wormhole IP routing filter(s) with an ATM switching network performs the same functions. IP packets are received by the wormhole routing filters; the bits of these packets do not arrive contiguously in time, but this is not a fundamental difference. The filter looks up the destination address of each packet in an IP routing table, just like a conventional IP router; the optimization here is that this packet address is contained in the first burst of bits (cell) that arrives at the filter. As a result of this routing lookup, an outgoing VP is chosen, and the packet is forwarded along this VP. This outgoing VP passes through an ATM switching network, and the latter directs all traffic through the VP to one of its output ports. The combined effect of a VP coice in the routing filter and of the ATM switching function is the same as in a conventional IP router: the packet is forwarded to a proper output port; the bits of the packet are not forwarded contiguously in time, but again this is an optimization rather than a fundamental difference. When output contention occurs, the ATM switches temporarily buffer the packets (or anyway the pieces of packets [cells] on which they operate).

As we see, wormhole IP routing filters with ATM switching networks are *equivalent* to conventional IP routers. Furthermore, the routing function for IP packets is carried out *completely in hardware,* in this wormhole IP over ATM system. Switching in the ATM (sub-) networks (figure 2) is performed in hardware, according to (permanent) VP's that have been set up (in software) ahead of time. The wormhole routing filters operate in hardware (section 4); their hardest function, routing lookup, is performed e.g. as in [GuLK98], within a delay of two DRAM accesses. Wormhole routing filters have a low delay (one or a few cell times), and can operate at multi-gigabit/second rates (section 4.3). The ATM architecture is also appropriate for building low latency and high throughput switching networks (see e.g. [KaSS97]). It follows that the wormhole IP over ATM architecture is an effective technique for building multi-gigabit-per-second IP routers that handle packets entirely in hardware; only set-up functions are performed in software, *off-line:* route selection (VP set-up), and routing table set-up.

Wormhole IP over ATM is not just "yet another technique" to build gigabit IP routers. Wormhole IP has a number of advantages, stemming from the very essence of its architecture. First, it allows the *integration* of IP and native ATM traffic, with both of them sharing the same transportation infrastructure, so that capacity left unused by one of them is exploited by the other. Quality of service (QoS) guarantees can be provided to native ATM traffic as usual, leaving such traffic unaffected by the added presence of IP packets. The road is also opened for IP to find new methods to take advantage of ATM's capabilities for QoS guarantees.

The second fundamental advantage of wormhole IP is that it *avoids the reassembly delay* at all intermediate routing nodes (all routing filters that are between ATM subnetworks). Packet reassembly delay can be very long (figures 22, 23), so avoiding it is important. In essence, wormhole IP performs *cut-through* routing, at the ATM cell level. One can argue that reassembly delay is long because of the spreading that the previous ATM subnetwork has caused (section 2.3); if all bits of each IP packet were forwarded contiguously in time, as in conventional IP routing, there would be no spreading, no reassembly delay, and cut-through would be less important. While this is true, remember that spreading is desirable so as to allow high priority traffic to *interrupt* long packet transmissions; this is precisely the method in which the network can offer QoS guarantees regarding delay. Also remember that the segmentation of variable-size packets into fixed-size units, and the *independent* switching of these units is the key technique used in building gigabit switches and routers.

While wormhole IP is a method to make IP routers that operate entirely in hardware at high speed, many other techniques have been proposed and used for routing IP over ATM. One group of such techniques is LAN emulation, classical IP over ATM, address resolution (NARP), next hop resolution (NHRP), and multiprotocol over ATM (MPOA). These techniques generally suffer from non-scalability when shared servers have to be used, and from connection set-up delays to send IP packets, since conventional ATM signaling is used. Another group is IP switching [NeML98] and its similar techniques, like IP/ATM, cell switch router (CSR), and ipsofacto. These techniques recognize *flows,* i.e. a sequence of packets with some common characteristics −primarily: destined to the same host. A flow is recognized after a (small) number of packets belonging to it have passed by. After recognition of a flow, an ATM connection is opened for that flow. Setting up such connections is performed using protocols faster than conventional ATM signaling, but still in software. Wormhole IP over ATM is faster than these methods, because it switches all packets −not just packets after their flow has been recognized− and because connections are set up in hardware, with negligible delay. This increased speed of wormhole IP does *not* require more expensive hardware. Finally, tag switching [RFC2105] and the similar ARIS proposal set up connections through an ATM (sub-) network ahead of time, and insert these into the IP routing tables, so that when IP packets arrive they can be forwarded immediately over these ready connections. This is similar to what wormhole IP does in the ATM subnetworks of figure 2. If the subnetworks over which connections have been pre-established are the same in tag switching and in wormhole IP, then the two techniques become comparable. In this case, wormhole IP introduces a new method for packets to be routed from one subnetwork to the other, in hardware, without packet reassembly and segmentation, hence with negligible delay. On the other hand, if tag switching tries to pre-establish connections end-to-end across a large ATM network, too many connections (labels) will be needed; if this results in preferentially pre-establishing some but not all connections, packets traveling on non-pre-established routes will suffer.

In conclusion, wormhole IP over ATM applies techniques from multiprocessor interconnection networks to making IP run over existing and future fast ATM equipment, at correspondingly high speed. Switching hardware prefers small, fixed-size traffic units. To be routed at high speed, IP packets are segmented into such fixed-size units (ATM cells). Once such a segmented packet enters the ATM world, it can be routed, according to its IP destination address, from subnetwork to subnetwork, fully in hardware, without intermediate reassembly, with negligible delays. This wormhole cell forwarding, from subnetwork to subnetwork, is performed by the routing filters proposed in this paper.

The wormhole IP routing filter that we proposed and analyzed has a number of advantages: *(i)* it works together with standard, existing ATM equipment; *(ii)* it allows the co-existence and integration of both IP and native ATM traffic in the same networks; *(iii)* the quality of service of native ATM traffic can stay unaffected by the added IP traffic, while IP can benefit from ATM's QoS capabilities; *(iv)* for IP traffic, the system operates equivalently to a network of low-latency gigabit IP routers. All these come at the modest cost of about 10 off-the-shelf chips per routing filter for 622 Mb/s operation; higher speed operation is feasible, even at 10 Gb/s, today. One factor contributing to simplicity is that, as we showed by simulation based on actual IP traces, a few tens of hardware-managed VC's per outgoing VP are sufficient; the bufferless organization is simpler, and performs very well.

Acknowledgements

We wish to thank the CS-534 class of Spring 1998, at the University of Crete, within which a lot of this work was carried out, and in particular Antonis Hatzistamatiou, Dimitrios Serpanos, and Georgios Stamoulis. We also thank Flavio Bonomi and Michael K. Wong for helping us with the literature, Vasilios Siris for his comments on a draft, and NLANR for making the IP traffic traces available.

References

[Barn97] R. Barnett (rbarn@westell.com): "Connectionless ATM", *IEE Electronics & Communications Engineering Journal,* Great Britain, October 1997, pp. 221-230.

[Bork90] S. Borkar e.a.: "Supporting Systolic and Memory Communication in iWarp", *Proc. of the 17th Int. Symp. on Computer Architecture,* ACM SIGARCH vol. 18, no. 2, June 1990, pp. 70-81.

[BCDP97] A. Brodnik, S. Carlsson, M. Degermark, S. Pink: "Small Forwarding Tables for Fast Routing Lookups", *Proceedings of the ACM SIGCOMM'97 Conference,* Cannes, France, Sept. 1997, pp. 3-14.

[DaSe87] W. Dally, C. Seitz: "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks", *IEEE Trans. on Computers,* vol. C-36, no. 5, May 1987, pp. 547-553.

[Dally92] W. Dally: "Virtual-Channel Flow Control", *IEEE Tran. on Parallel and Distributed Systems,* vol. 3, no. 2, March 1992, pp. 194-205.

[Gall97] M. Galles: "Spider: A High-Speed Network Interconnect", *IEEE Micro,* vol. 17, no. 1, Jan./Feb. 1997, pp. 34-39.

[GuLK98] P. Gupta, S. Lin, N. McKeown: "Routing Lookups in Hardware at Memory Access Speeds", *Proceedings of IEEE Infocom,* San Francisco, CA USA, April 1998; http://tiny-tera.stanford.edu/~nickm/papers/ Infocom98_lookup.ps or Infocom98_lookup.pdf

[HaSt98] A. Hatzistamatiou, G. Stamoulis; personal communication, CS-534 course, University of Crete at Heraklion, Spring 1998.

[KaSS97] M. Katevenis, D. Serpanos, E. Spyridakis: "Switching Fabrics with Internal Backpressure using the ATLAS I Single-Chip ATM Switch", *Proceedings of the GLOBECOM'97 Conference,* Phoenix, AZ USA, Nov. 1997, pp. 242-246; http://archvlsi.ics.forth.gr/atlasI/atlasI_globecom97.ps.gz

[LeBo92] J. LeBoudec: "The Asynchronous Transfer Mode: A Tutorial", *Computer Networks and ISDN Systems,* vol. 24, no. 4, May 1992.

[NeML98] P. Newman, G. Minshall, T. Lyon: "IP Switching: ATM Under IP", *IEEE/ACM Transactions on Networking,* vol. 6, no. 2, April 1998, pp. 117-129; http://www.ipsilon.com/~pn/papers/ ton97.ps or ton97.pdf

[NLANR97] USA National Laboratory for Applied Network Research (NLANR) (http://www.nlanr.net): IP traffic trace from FIXWEST West Coast federal interexchange node of January 1997, available at: ftp://oceana.nlanr.net/Traces/FR+/9701/

[RFC2105] Y. Rekhter, B. Davie, D. Katz, E. Rosen, G. Swallow: "Cisco Systems' Tag Switching Architecture Overview", IETF RFC2105, Feb. 1997; http://info.internet.isi.edu:80/in-notes/rfc/files/rfc2105.txt

[WVTP97] M. Waldvogel, G. Varghese, J.Turner, B. Plattner: "Scalable High-Speed IP Routing Lookups", *Proceedings of the ACM SIGCOMM'97 Conference,* Cannes, France, Sept. 1997, pp. 25-36.