

Supporting TCP connections in wormhole routing and ATM networks

A. Bianco, E. Leonardi, M. Munafò, F. Neri*

Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy

Received 6 November 2000; accepted 6 November 2000

Abstract

An interesting lightweight switching technique for local area and campus environments is wormhole routing, in which the head of a packet, upon arriving at an intermediate switch, is immediately forwarded to the next switch on the path. The current proposals and products of wormhole routing networks provide a connectionless and lossless transport of variable-size data units, aiming at minimal latencies, but without any quality-of-service guarantee. Wormhole networks therefore push at an extreme the networking paradigm of the Internet, adopting technical solutions for high-speed networking that are opposite to those taken in ATM.

The aim of this paper is to perform a simulation-based comparison between the wormhole routing and ATM transport techniques in high-speed networking scenarios. The behavior of the two information transport techniques is studied in mesh topologies where a few interfering TCP connections in overload interact with a uniform background traffic.

Our results show that a larger hardware and software complexity is required to ATM switches with respect to wormhole routing switches in order to provide a comparable throughput to TCP connections. By contrast, ATM is more capable to handle severe congestion situations and to provide better fairness. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: TCP; ATM; Wormhole routing; Deflection routing; Backpressure

1. Introduction

ATM and wormhole routing are two data transport techniques suited for high-performance networking.

Wormhole routing is a lightweight protocol and switching technique for local and campus high-speed data networks based on mesh topologies. With respect to the conventional store-and-forward packet switching, in wormhole routing the head of a packet (worm), upon arriving at an intermediate node, is immediately forwarded to the next node on the path, without waiting for the entire packet to be assembled [1]. Thus, the packet, like a worm, may stretch across several intermediate switches and links.

An important example of wormhole communication network is the crossbar-based Myrinet [2,3], which uses back-pressure flow control, and source routing.

The main goals of wormhole routing networks are simplicity (i.e. reduced switch complexity) and low latency. They are usually operated in the local area at very high speeds in an asynchronous and unslotted fashion. Since losses are very costly in networks with large delay \times bandwidth

products, wormhole networks are designed as loss-free systems: worms are never discarded upon congestion.

In order to provide low latency and permit simple switches, wormhole switches are equipped with very limited buffering capacities. Hence, a careful flow control must be exercised to avoid buffer overflows. Moreover, the choice of avoiding worm losses makes wormhole networks particularly prone to deadlocks or livelocks, and to congestion, so that congestion control techniques are required. Traditionally, flow control is obtained with backpressure: blocked worms, i.e. worms that cannot proceed towards their destination due to lack of resources, are frozen in place using explicit node-to-node signaling. With backpressure flow control, deadlocks are prevented either by restricted routing (usually Up/Down routing on a spanning tree), or by partitioning buffering and transmission resources (we consider shortest path routing with virtual channels [4]). Asynchronous (and unslotted) deflection routing with local input control has been proposed [5] as a possible alternative to the backpressure mechanism. Livelocks are possible with asynchronous deflection routing; they can be avoided by means of suitable (random) worm alignment techniques at switches.

ATM has been considered as a very promising transport technique for high-speed networks, both in the technical

* Corresponding author. Tel.: +39-11-564-4076; fax: +39-11-564-4099.

E-mail addresses: bianco@polito.it (A. Bianco), leonardi@polito.it (E. Leonardi), munaf@polito.it (M. Munafò), neri@polito.it (F. Neri).

Table 1
Comparison of wormhole routing and ATM

Feature	ATM	Wormhole
Small switching delays via	Cell pipelining	Cut-through switching
Segmentation overhead	Yes	No
Losses of data units	Possible	Non-permitted
Error control	Possible	No
Virtual circuits	Yes	No
QoS guarantee	Good	Poor
Memory requirements	Large	Small
Costs	Large	Small
Scalability	Good	Poor

community, and among manufacturers and operators. Although the interest in ATM technology has recently decreased mainly due to the Internet exponential growth, ATM still plays an important role in backbones and in the wide area when QoS guarantees are a major design goal. Questions arise on the suitability of ATM in local environments, mainly because of complexity and costs.

The technical solutions adopted in ATM are very different from those of wormhole networks. Wormhole routing networks do not require segmentation and reassembly of data units, since worms are variable in size. Moreover, while ATM cells can be discarded, worms are never dropped upon congestion. Furthermore, wormhole networks normally take the connectionless approach, not relying on virtual circuits and call admission control. Table 1 contrasts the wormhole routing and ATM transport techniques.

Besides costs, which remain high, a potential limitation of ATM in supercomputer interconnect applications and cluster computing is the high latency, due to bandwidth allocation/negotiation at call set up (which can indeed be dealt with using permanent connections) and cell segmentation/reassembly during the data transfer phase.

Wormhole routing networks are well suited for the transport of datagram traffic when low latency is important. ATM is best suited to transport heavy traffic flows with quality of service requirements, like in the case of multimedia applications. Several studies have been performed on the network-level performance of wormhole routing networks [5] and of ATM [6,7]. It is, however, interesting to study the effects of the information transport techniques on higher level, end-to-end protocols, like TCP; this problem was only partially addressed up to now. TCP [8,9] (the Transmission Control Protocol of Internet) stands as *the* natural candidate, being today the most widely used end-to-end protocol in packet networks. It is currently used in the Internet to support a wide range of applications with different service requirements, although it is mainly used in the context of data traffic.

In this paper we present a simulation study of the performance of TCP connections in simple network configurations, taking as performance indices the normalized TCP goodput, i.e. the throughput at the receiver, discarding all retransmitted packets, divided by the link speed, and the

efficiency, i.e. the ratio between the goodput and the total offered load of TCP connections. The study is done for wormhole routing networks, with both backpressure and deflection flow control, and for ATM networks. We carry on the comparison keeping the two systems (wormhole and ATM) as similar as possible in terms of topology, offered traffic, switch architecture, and parameters like link speed and buffer sizes. We study the performance of TCP over ATM connections with UBR (Unspecified Bit Rate) traffic, and with two ABR (Available Bit Rate) transfer capabilities: a simple RRM (Relative Rate Marking) [10] and a more sophisticated ER (Explicit Rate) scheme [11]. The former is a simple control algorithm, which relies on the node output buffer occupancy and its derivative as indicators of network congestion and provides sources with a ternary feedback (keep, increase and decrease rate). The latter is a well-known algorithm, which monitors the node load on each output link and advises sources about the rates at which they should transmit.

Several studies of the behavior of TCP in ATM networks are available in the literature [12–17]. We do not address here the problems arising from the interaction between IP (the connectionless Internet Protocol) and ATM that has received large attention in the recent past [18,19], as we mainly focus on local area or campus networks, where the routing functionalities of IP play a minor role.

The paper is organized in two main sections. Section 2 describes the rules of the game of our simulation based performance comparison. Our modeling of TCP connections, of ATM nodes with ABR, and of wormhole networks with backpressure or deflection flow control is described, together with the selected topologies and traffic scenarios. Section 3 brings the two transport techniques in the arena, presenting and discussing selected simulation results.

2. Rules of the game

2.1. TCP model

We included in our simulation programs (both for ATM and for wormhole networks) the same detailed model of the BSD 4.3-reno version of TCP. The model was obtained [12] by stripping, with the necessary modifications, parts of the code from a real implementation of TCP.

Given that TCP is a well-known protocol, we briefly outline here only the main characteristics of the TCP-reno congestion control algorithm. Further details can be found in Refs. [20,21].

TCP is a connection-oriented protocol based on a window mechanism which regulates the flow of data units. The window size at the transmitter is dynamically incremented until either the congestion is detected into the network, or a maximum window size, advertised by the receiver, is reached. The receiver returns acknowledgments (ACKs) of the received data (piggybacking it onto data segments

if possible); receiving (or not receiving) ACKs at the transmitter triggers either the transmission of a new segment (if available) and the increase of the window, or the beginning of congestion control procedures.

At the transmitter, a threshold is initially set to the maximum receiver-advertised window size, and the actual window size is set to one segment. The window increases in two phases: the *slow-start* (or *congestion recovery*) phase, where the window grows exponentially, and the *congestion avoidance* phase, where the window increases linearly. The transmitter goes from the slow-start phase to the congestion avoidance phase when the actual window size reaches the current threshold. When the maximum window size is reached, the transmission continues without further increasing the window size.

The source considers the loss of a segment as a symptom of congestion, and reacts reducing the transmission window size. The segment loss can be detected either through a timer expiration (the ACK is not received within a defined delay from the segment transmission), or by the *fast retransmit with fast recovery* algorithm. If a retransmission timer expires, the transmission window size is reduced to one segment, the threshold is set to half the current window size and the slow-start phase is entered. Using the fast retransmission procedure, the source node infers that a segment was lost if four ACKs referring to the same segment are received. Owing to the fast recovery procedure the source reacts by halving the current transmission window size, and switching to the congestion avoidance phase by setting the threshold to the current transmission window size.

Since the BSD 4.3-reno TCP is not tailored to high-speed networks [22], slight modifications were required to adapt its code to the high-speed environment; the most important aspects of this adaptation process are described below. We included practically all the important TCP features, with the exception of the selective ACK [23] and the delayed ACK options [24].

An important aspect in TCP implementations is the timeout setting and the RTT (Round Trip Time) estimation. Timeouts in TCP are computed as a smoothed estimate based on the RTT average μ_{RTT} , and standard deviation σ_{RTT} , as $t_0 = \mu_{\text{RTT}} + 4\sigma_{\text{RTT}}$; the average RTT and its standard deviation are computed following the algorithm proposed by Van Jacobson [20]. In the present implementations of TCP, the transmitter timers evolve with a *granularity* (the minimum value to which a timeout can be set) that is rather coarse (of the order of 200 ms); this granularity should be reduced for the protocol to react fast on networks with short round-trip delays. In our simulation setup, the user can choose the timer granularity of the TCP implementation. As a rule of thumb, the granularity should be smaller than the propagation delay along the connection, and it should be equal for all connections, if large throughput and, more important, fairness among connections sharing the same resources has to be achieved.

2.2. ATM and ABR

The ATM simulation program was implemented in CLASS [25], a package for the cell-level simulation of ATM networks developed at Politecnico di Torino, under a contract with CSELT.

Network nodes in CLASS are ideal output-buffer ATM switches. The switching matrix is non-blocking and can transfer to output ports in one slot time all the cells received at the inputs. All the available buffer space is used at output ports.

CLASS provides several traffic control algorithms, specially for the ABR service. We consider a simpler RRM (Relative Rate Marking) operating mode, where the network feedback can only assume three values, and a more complex ERM (Explicit Rate Marking) operating mode, where the network explicitly notifies sources about their assigned cell transmission rates. In both schemes, the key issue that determines the performance of the network is the control algorithm implemented within the ATM switches to decide as to what feedback must be returned to sources.

The ABR RRM control algorithm we use is inspired by standard techniques in control theory, and designed aiming at simple implementations in low-cost ATM switches [10]. The algorithm relies on the periodic measurement of the buffer occupancy and its derivative, which triggers the congestion detection and the ABR feedback to sources. More details are provided below.

The considered ABR ERM mechanism is ERICA (Explicit Rate Indication for Congestion Avoidance), proposed in Ref. [11]. The algorithm requires a periodic measurement of both the available bandwidth capacity for ABR traffic and the number of currently active VCs. The cell transmission rates explicitly assigned to sources are determined according to a fair share. If some VC is not using its fair share, the remaining capacity is allocated fairly among other VCs. The goals of the algorithm are two-fold: fully (and fairly) exploiting the bandwidth available for ABR traffic, and keeping the output buffer occupancy very close to a given level, defined via a parameter called as *target delay* in this paper. The details of the algorithm are quite complex and out of the scope of this paper; the interested reader is referred to Ref. [11] for a complete description. CLASS implements the full algorithm proposed in Ref. [11], with the exclusion of the filter function.

The ABR connections that we consider carry a sustained TCP traffic, which can be interpreted as being generated by sources performing long file transfers using the TCP protocol. The performance of TCP connections supporting file transfers over ATM networks was already studied by several authors [12–17]. Results indicated that when the UBR service category (i.e. no traffic control) is used, performance is generally rather poor, mainly due to the TCP protocol dynamics, which cannot be easily adapted to networks with large delay \times bandwidth products [22]. When an RRM ABR scheme with a very simple control

Table 2
ABR parameter values used in the simulation runs

Algorithm	Parameter	Value
RRM and ERM	PCR (Peak Cell Rate)	640 Mbit/s
	MCR (Minimum Cell Rate)	1 Mbit/s
	ICR (Initial Cell Rate)	150 Mbit/s
	TBE (Transient Buffer Exposure)	512 cells per VC
RRM	RIF (Rate Increase Factor)	1/64
	RDF (Rate Decrease Factor)	1/16
	β	15 cells
	l_t	10 cells
	h_t	200 cells for buffer 2000 50 cells for buffer 200
	Measurement interval	50 cells
ERM	Target delay	500 cells for buffer size 2000 50 cells for buffer size 200
		Measurement interval S_r

mechanism is instead used in the ATM network to support TCP connections, satisfactory performance can be achieved, provided that the ABR parameters are carefully tuned. Better results are obtained when ERM algorithms are adopted.

2.2.1. More details on the ABR RRM scheme

In the RRM ABR scheme, feedback to sources is conveyed by the network through the NI (No Increase) and CI (Congestion Indication) bits of the RM (Resource Management) cells. The feedback indication can only assume three values corresponding to ‘increase the cell transmission rate’ (NI = CI = 0), ‘keep the present cell transmission rate’ (NI = 1, CI = 0), or ‘decrease the cell transmission rate’ (CI = 1, NI not significant). Sources react to these feedback indications by modifying their cell transmission rates according to their values of the ABR parameters RIF (Rate Increase Factor) and RDF (Rate Decrease Factor), which are negotiated at connection setup.

The NI and CI bits can be set to 1 by ATM switches along the connection path, depending on the switch internal congestion. We focus on the algorithms to set the NI and CI bits, and we assume that they are based on the occupancy of the link buffer (at output ports of switches), which is shared by the ABR connections and the background traffic.

Since the core algorithm on which the RRM ABR scheme is based is very simple, it can be expected that RRM schemes will be implemented in low-cost ATM switches; as a consequence, also the control algorithms implemented within the ATM switches must be very simple. These remarks about the type of ATM switches that we consider lead to the assumption that no sophisticated queuing and traffic control mechanisms are available within the switches: we therefore assume that switches have no per-VC or per-traffic-class separate buffering, and they are not able to compute the instantaneous load offered by the connections.

The only available information on link congestion is hence the buffer occupancy Q .

The goal of the control of the system is three-fold: (i) avoid losses in the buffer; (ii) maximize the link utilization; and (iii) minimize the buffer occupancy fluctuations to avoid introducing unwanted jitter in the cell delay (in our scenario the buffer is shared by ABR and background traffic, the latter possibly being delay sensitive).

We adopt a control algorithm called PD +, ‘P’ standing for position (or level) of the buffer occupancy and ‘D’ for its derivative; more details can be found in Ref. [10]. The buffer occupancy is measured at regular epochs, S_r slots apart; between two consecutive samplings the control signals are kept constant.

Let $D_Q(t_n) = Q(t_n) - Q(t_{n-1})$ be the two-point derivative of the buffer occupancy Q , where t_n is the n th sampling instant. This technique gives great importance to buffer occupancy variations: if $|D_Q(t_n)| > \beta$ then the value of Q is ignored. The goal of these modifications is to detect important traffic variations as soon as possible and, as a consequence, to keep the buffer as empty as possible to minimize cell losses. The mapping of the values of Q and D_Q over the control feedback is the following:

$$\forall Q; D_Q < -\beta \Rightarrow \text{increase rate}$$

$$\forall Q; \beta < D_Q \Rightarrow \text{decrease rate}$$

$$Q < l_t; -\beta \leq D_Q \leq 0 \Rightarrow \text{increase rate}$$

$$Q < l_t; 0 < D_Q \leq \beta \Rightarrow \text{keep rate}$$

$$l_t \leq Q < h_t; -\beta \leq D_Q \leq \beta \Rightarrow \text{keep rate}$$

$$h_t \leq Q; -\beta \leq D_Q < 0 \Rightarrow \text{keep rate}$$

$$h_t \leq Q; 0 \leq D_Q \leq \beta \Rightarrow \text{decrease rate}$$

The parameter β , together with the two thresholds l_t and h_t , can be used to tune the control algorithm.

2.2.2. Selected parameters

Both the RRM and the ERM ABR implementations in CLASS conforms to the ATM Forum Traffic Management Specification 4.0 [26]. Nodes monitor the status of the buffer in the forward direction (i.e. on the link going from the transmitter to the receiver) and set the feedback indication on the RM cells flowing in the backward direction (i.e. on the RM cells going from the receiver to the transmitter).

The ABR parameters setup is kept constant in all the simulations: the most significant parameter values are summarized in Table 2. ICR is the rate at which sources start to transmit after a long silence period, PCR the maximum allowed transmission rate, MCR the minimum guaranteed transmission rate, RIF the additive rate increase factor, RDF the multiplicative rate decrease factor and TBE is the transient buffer exposure, i.e. the number of cells that can be transmitted at connection start-up before receiving the first RM cell. The TBE value is selected so as to accommodate all the cells transmitted by the source at ICR during a round trip time.

We do not claim that the chosen parameter vector is the best possible choice for the considered situation, but the values are reasonable (we are not interested here in finding the best possible ABR parameters combination). However, the conclusions we draw are not affected by the specific choice of the algorithm parameters.

2.3. Wormhole routing

The simulation programs for wormhole routing were implemented using MAISIE [27], a package built as an extension on the C language for the object-oriented discrete-event parallel simulation. The basic simulation engine chosen for wormhole is different with respect to the ATM simulator due to the asynchronous nature of the protocol. To make a meaningful comparison, the models of the traffic sources (including the code of the TCP model) were however the same for the two simulators.

2.3.1. Wormhole with backpressure flow control

The first considered flow control scheme is backpressure, which is most common in wormhole LANs. When a worm arrives at a node, it has a choice of output ports through which it can reach the destination in the fewest hops. These ports are called preferred output ports. With backpressure flow control, the worm is routed to an unused port in this set. If this output port is not available, the worm is blocked, using a backpressure signal on the return link. When congestion occurs in a backpressure LAN, blocked worms prevent other worms from reaching the congestion point.

Backpressure is therefore an explicit node-to-node flow control mechanism that requires the bidirectionality of network links. A fixed size buffer, called slack buffer, is

associated with each receiver in the nodes. STOP and GO flow-control signals at the physical layer are used to regulate the flow of traffic in both directions across a link. They are sent in the direction opposite to the one on which the traffic is being received (thus, the need for bidirectional links). Suitable high-water and low-water marks in the slack buffer are used to decide when to send a STOP signal and when to send a GO signal, respectively. The size of the slack buffer is dependent on the link round-trip propagation delay.

With backpressure, worms may thus be blocked on their way to the destination if no resources are available along the path. As circular waits are possible, backpressure is prone to deadlocks [4,28]. The issue of deadlocks in computer systems has been well studied [29–31]. In wormhole routing networks, deadlocks are more likely than in conventional store-and-forward networks due to the fact that long worms need the progressive allocation of several resources (links and buffers) to be transported to their destination. Although one or more worms could be dropped upon detection of a deadlock, deadlock avoidance is preferable in high-speed networks due to the large delay \times bandwidth product.

One popular approach to avoid deadlocks is to restrict the routes that a message may take towards the destination. Finding and using routes in the network in such a fashion that deadlocks are avoided works at the cost of non-optimal routes (in terms of number of hops). This approach was adopted in the Autonet [32], and in Myrinet [2,3]. In such networks, a spanning tree is constructed on the topology, and all hosts use this tree to determine the routes to all other hosts. This prevents circular waits and guarantees deadlock freedom of the routes at the cost of non-minimum-distance routing, and of performance degradation due to congestion around the root of the tree.

Another well-known approach in store-and-forward packet networks to avoid deadlocks consists of defining a partial order on the switch buffers (hierarchy of packet buffer pools) at each node [30,33–36], and allocating them on various criteria like number of hops traveled, the route which the packet is traveling, etc. An approach similar to the hierarchy of buffer pools, called the *virtual channels* technique [4], is used in our simulations. It combines restricted routing and buffer partitioning to achieve deadlock-free, minimum-distance routing in the wormhole routing context.

Virtual channels are groups of channels that share a physical channel. Each virtual channel has its own slack buffer and flow-control logic. Thus, the buffer space is statically partitioned according to the number of virtual channels, while the bandwidth available on the physical channel can be dynamically allocated to virtual channels. Physical channels that belong to a cycle in the network are decomposed into a group of virtual channels, and the routing is restricted only in the sense that the proper virtual channel must be selected when a physical link must be crossed. The concept of virtual channels implies a protocol to serve the virtual channels that are mapped onto the same physical

link. Protocols of this sort were presented in Ref. [37]. We consider in this paper the Non-preemptive Priority (NP) protocol: priority is given to worms using higher numbered virtual channels since they are closer to the destination. A worm is served until either it has been completely transmitted or it is blocked. In either case a worm in the highest virtual channel which can be served is served next.

Virtual channels allow minimum distance routing, but require an increase in the hardware complexity of the node: the input buffer and the flow control signaling and logic must be replicated for each virtual channel.

2.3.2. Wormhole with deflection flow control

Another technique which can be used to limit congestion and prevent packet loss in worm LANs is deflection routing. The scheme works as follows: if the worm arriving at the switch finds its preferred output (as defined by a routing table or by a source routing header) busy, it is ‘deflected’ to any other output which happens to be free.

The first deflection routing studies [38–40] were aimed at synchronous, slotted networks. Deflection routing is possible on any regular topology (i.e. with a number of input ports at each switch equal to the number of output ports). Deflection routing is not affected by the problem of worm loss due to buffer overflow, since there is always an output port available (not necessarily among the preferred ones) for the worm to be transmitted out.

Flow control in deflection routing is achieved jointly by deflecting worms away from the congested area and by controlling the input of traffic from the hosts. We call the latter technique *input rate control*. Flow control in deflection routing is triggered by the implicit propagation of information about congestion in the network: deflections spread the congestion to the other nodes in the network until the hosts are throttled via input rate control.

2.3.2.1. Livelocks. Deadlocks are not possible when deflection routing is used since worms are never blocked. However, livelocks are possible. A livelock occurs when worms are endlessly deflected in a circle and never reach the destination. To overcome this problem, we consider the following worm alignment (or *delayed deflection*) technique. When the head of a worm arrives at a node and the preferred output port is not available, a calculation is made to see if the worm can be buffered in the input buffer until one of the preferred output ports becomes available (i.e. till the remainder of the worm using that port is transmitted). If the buffer would overflow, then the worm is deflected through another output port. Otherwise, the worm is buffered and deflection is delayed. It is possible that, by the time the output port becomes available, other worms waiting for the same output port may have arrived. One of the waiting worms is randomly chosen and the output port assigned to it. This worm alignment mechanism introduces more choices of output port selection, thereby improving the overall efficiency. It can

also significantly reduce the probability of livelocks (which were never observed in our simulation experiments).

2.3.2.2. Routing. Routing decisions at a node are based on the weights associated with each output port according to the distance from the destination, minimum weight indicating a preferred port. For each incoming worm, all preferred output ports are scanned to find a free port. If none is found (and the worm should be deflected), deflection is delayed using the worm alignment protocol described earlier. In case of deflection, minimum-weight free ports are considered first.

The ports of a node in the network are designated as either host ports (connected to hosts) or transit ports (connected to other nodes). The buffers associated with the input ports are accordingly called either host buffers or transit buffers. Whenever an output becomes free, all input ports are searched for worms awaiting transmission. The order in which input ports are scanned can significantly affect performance, especially in our scenarios comprising overloaded connections. We assume that input ports are scanned in a random order, giving priority to transit ports over host ports (all transit ports are scanned first). In Myrinet a round-robin scan of all input ports is implemented to limit complexity.

Since we assume that hosts can receive only one worm at a time, deflection may occur also at the destination: if the output port associated with the destination host is busy, an arriving worm may be deflected away from the destination.

2.3.2.3. Buffering. Wormhole routing switches are input buffered, both in the backpressure and in the deflection cases.

The input buffer requirements for deflection routing are determined by the worm alignment protocol and by the procedure for injection of worms into the network by the local host. While the local worm is being transmitted, it is possible that a ‘transit’ worm arrives and finds that all the ports of the switch are in use, except for the output port to the host. Although other solutions are possible, we assume in this paper for simplicity that transit worms are stored in the transit buffers when the host starts transmission of a new worm. To avoid overflow of transit buffers, transmission of the host worm is enabled only when all transit buffers have a free space at least equal to the size of the new worm. This implies that the size of the input buffer is constrained by the maximum worm size.

2.3.2.4. Input rate control. Deflection routing suffers from two major drawbacks: out-of-order delivery and unbounded delays. For what regards the first problem, in Ref. [41] the authors mention several applications (like file transfers) for which in-sequence delivery of messages is not essential. We will observe later in the paper that TCP connections do not degrade their performance with deflection routing, hence they well tolerate out-of-order worm delivery. This is mainly due to the tight access control induced at the

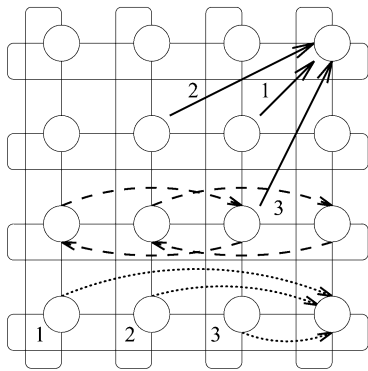


Fig. 1. The three traffic scenarios.

source by the algorithms adopted in wormhole networks, which produces behavior largely independent from the TCP maximum window size. As shown in Ref. [5], the problem of unbounded delays can be effectively mitigated by input rate control at the hosts. A time-to-live mechanism, which eliminates ‘unlucky’ worms after a given time or hop-count, can help both to clip the tail of the delay distribution, and to eliminate residual livelock problems, specially in our context in which the TCP flow control can recover worm losses.

We assume that host worms are never deflected. Accordingly, the first simple input control policy, which is usually used in our simulation experiments, requires the host worms to stay in the host buffer until a preferred output port is free and a transit worm is not waiting on it. Further, the transmission of a host worm does not start if the amount of free space in each transit buffer is less than the worm size (to avoid buffer overflow). Although this approach seems to be very restrictive with respect to the admittance of host worms into the network, it is not. Simulation results have shown that the host may still be able to overload the network, causing an increase of deflection and, as a consequence, degraded performance.

2.4. Topologies, traffic and node architecture

The mesh topologies used in our simulation experiments are 16- and a 64-nodes Manhattan networks, i.e. square grids wrapped around a torus, with node-to-node link lengths equal to 1 km and link speed equal to 640 Mbit/s. Mesh topologies were chosen to keep in the scenario deflection routing; however, observe that the traffic patterns chosen for TCP connections described later are representative of widely used topologies, normally considered as good abstractions of real topologies: the bottleneck and the parking lot topology. All network nodes act both as switches, and as information sources and sinks. A uniform Poisson background traffic with varying intensity is considered in all simulations, and the total load on the network is taken as a simulation parameter. The Poisson characteriza-

tion was taken for simplicity, since background traffic is mainly used to break synchronization problems among the TCP connections, whose behavior would otherwise be deterministic.

At each node, each port is equipped with a packet buffer whose size B is at least slightly larger than the TCP Maximum Segment Size (MSS), which we assume to be equal to 9140 bytes. For the input-buffered wormhole routing, the case $B = 10\,000$ bytes is always considered; for ATM, ideal output buffered switches are considered, with either small ($B = 200$ cells, i.e. $B = 10\,600$ bytes), or large buffers ($B = 2000$ cells, i.e. $B = 106\,000$ bytes). The IP protocol is taken into account only in packet headers (the TCP PDU is augmented by the standard size of the IP header), but no functionalities of the protocol are simulated, as mentioned previously.

We consider maximum window sizes W for TCP connections equal to 1, 5, and 30 segments; a value of $W = 5$ segments is large enough to allow any possible TCP connection to obtain a 640 Mbit/s throughput in our (small) networks.

The timer granularity was set to $5\ \mu\text{s}$ due to the small network size. Simulations with higher granularity (50 ms) have produced very similar results.

We consider three different traffic patterns for TCP connections. In all scenarios, few overloaded TCP connections are competing with the background traffic for network resources.

In the first scenario, three TCP flows interact in a 16-nodes grid: they are depicted in Fig. 1 with solid lines. The three receivers share the same destination node, and the three transmitters are numbered in the figure. One of the TCP sources (source 1) is positioned so that it suffers from the traffic generated by the two other TCP sources (sources 2 and 3); TCP sources 2 and 3 are connected to nodes in topologically equivalent positions with respect to the destination node.

In the second scenario, four TCP connections lie on a horizontal ring of a 16-nodes Manhattan network, so that connections share, pairwise, a common link, as depicted with dashed lines in Fig. 1. The symmetrical positions along the ring should not privilege any of the TCP connections.

In the third scenario, we embed a ‘parking lot’ topology with three TCP transmitters on a ring of a 64-nodes Manhattan network; the third TCP connection suffers from the traffic generated by the other two TCP connections. The dotted lines in Fig. 1 depict the relative positions of the three TCP flows. A 64-nodes grid was considered in this case (instead of a 16-nodes grid) to obtain overlapping minimum-distance routes.

The routes taken by packets belonging to the three TCP connections in the first traffic scenario will either be disjoint or overlapping, whereas in the two other scenarios only overlapping routes are considered to put in evidence congestion phenomena. These routes are decided at

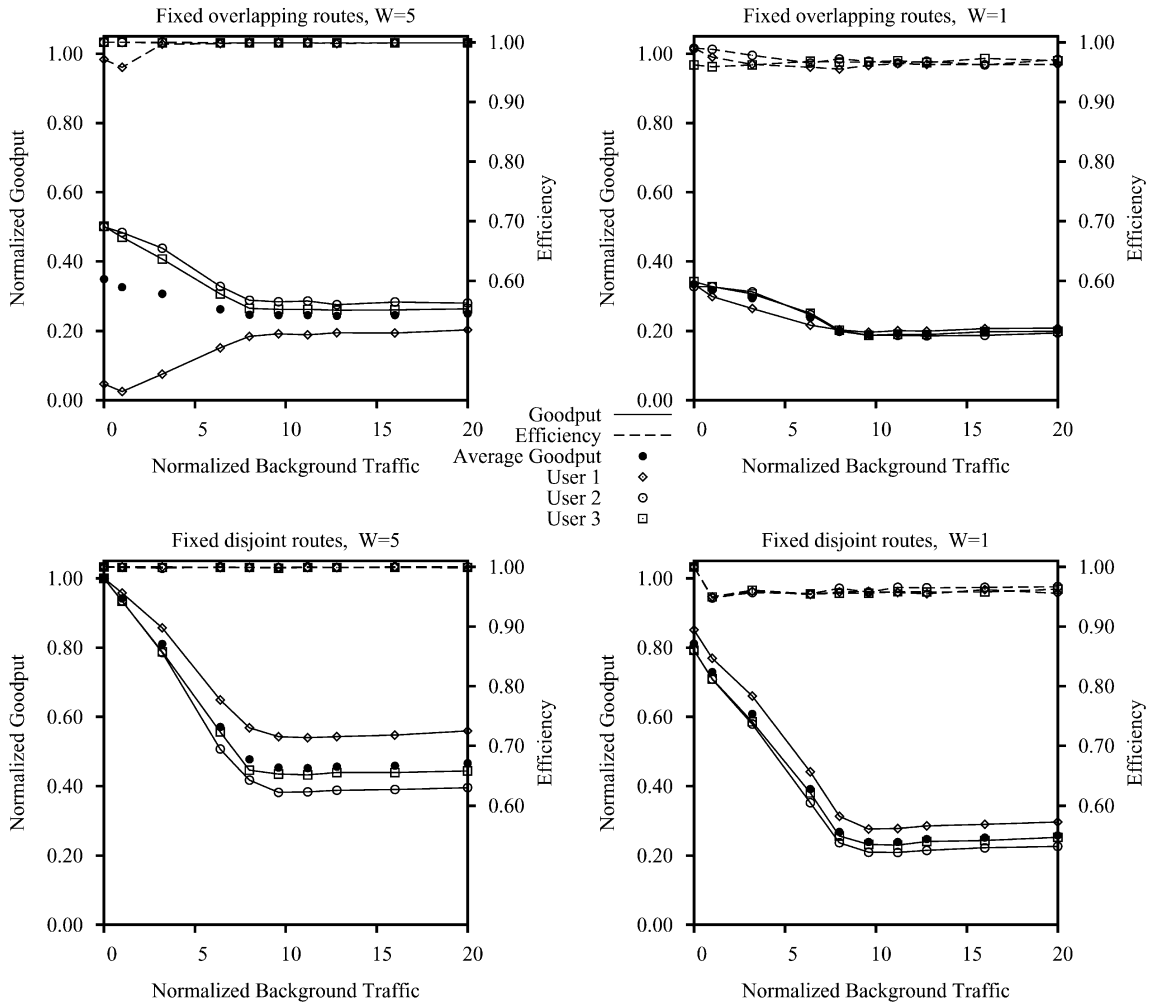


Fig. 2. Goodput and efficiency for wormhole routing with backpressure flow control. Medium ($W = 5$) and small ($W = 1$) window sizes. Overlapping and disjoint fixed routes.

call set-up time for ATM (and in our simulation they are an input parameter). For wormhole with backpressure, routes can either be fixed, or adaptive. In the former case an incoming worm can be routed only to one output port (this is the case of the source-routed Myrinet); while in the latter case any unused minimum-cost output port can be selected by an incoming worm. In the case of deflection, instead, packets travel freely in the topology.

ATM network nodes are output-buffer switches with FIFO queue discipline. We consider a single FIFO queue in all scenarios to keep ATM switches as close as possible to wormhole switches. This means that ABR (TCP) traffic and background traffic share a common buffer. It must however be noted that ABR control techniques in general assume more complex, separate queuing scenarios, which should clearly lead to better performance. This is one of the reasons why some of the performance obtained with ABR controls are less satisfactory than expected.

3. Performance results

The performance indices considered in the paper are the TCP goodput, i.e. the throughput at the receiver discarding all retransmitted packets, and the efficiency, i.e. the ratio between the goodput and the total offered load of TCP connections. Note that the efficiency index is, in the loss-free wormhole networks, an indicator of retransmissions induced by the variations of the connection round trip time, whereas in ATM network is (mainly) an indirect measure of cell losses due to buffer overflows.

3.1. First traffic scenario

The results when backpressure flow control is applied together with virtual channels on fixed overlapping routes are reported in the upper part of Fig. 2, and show that unfairness arises among TCP connections with window size equal to 5. This is induced by the topological position of source 1

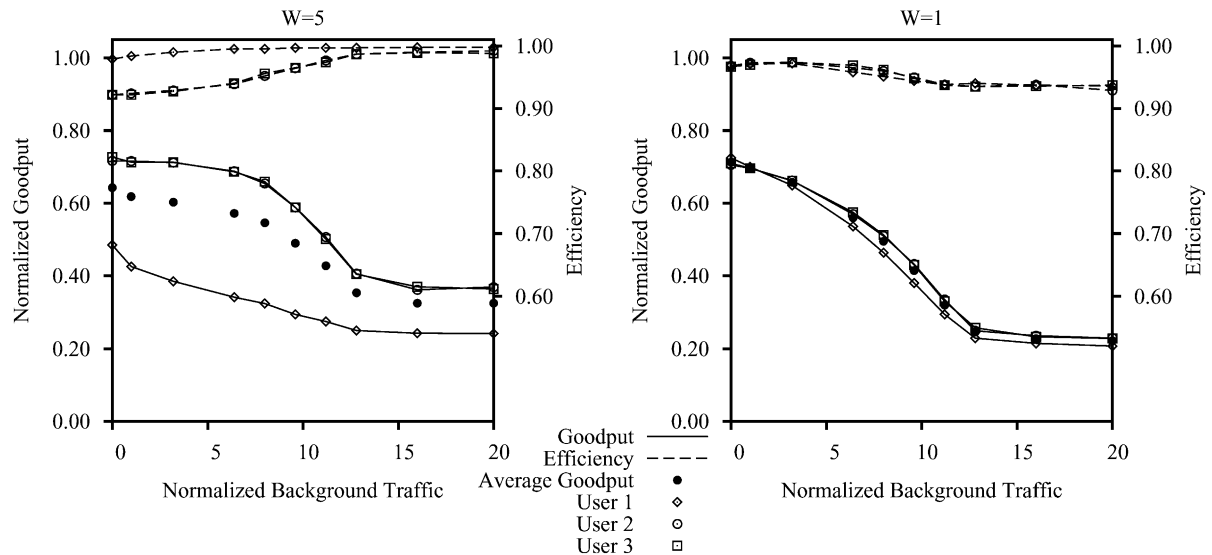


Fig. 3. Goodput and efficiency for wormhole routing with deflection flow control. Medium ($W = 5$) and small ($W = 1$) window sizes.

together with the priority given to in-transit worms with respect to locally generated worms. The minor unfairness among the two other connections, which are situated in topologically equivalent positions, is induced by the adopted virtual channel mechanism: among all minimum distance routes, only those with one horizontal path followed by one vertical path are permitted [4,28]. Under this constraint, routes overlap in our scenario, causing interference. All connections follow the horizontal path before the vertical one. Consequently source 2 path completely comprises source 1 path, with two hops along the east direction in the grid and one along the north direction (see Fig. 1); while source 3 path overlaps the other two in the final north link, which is therefore shared by all three sources. The average throughput for TCP connections ranges from 0.35 when no background traffic is present to a limit of about 0.2 when the network is overloaded.

When the window size is reduced to one, better fairness is observed. Although unit windows are able to fully load network links, given the short distances between TCP sources and destinations, they prove to be less tolerant to errors in the RTT estimation: retransmissions occur (note the reduced efficiency), and link utilizations are below 100%. As a consequence, downstream sources also gain a fair share of the available bandwidth.

In the lower part of Fig. 2 the case of disjoint fixed routes is presented. With respect to the real world, this is an optimistic situation, which minimizes the interference between the three TCP connections. Indeed, performance is much better: a doubled goodput is observed, specially for the case $W = 5$, which does not throttle TCP sources. Unfairness is reversed: source 1 can now take advantage of the shorter distance from the destination.

A certain degree of unfairness is observed among the three TCP connections in all the graphs of Fig. 2. This is typical of wormhole networks, where traffic is less

controlled with respect to ATM networks, and it will be observed in subsequent results, particularly when congested traffic patterns are present (see Figs. 11 and 12).

Fig. 3 considers deflection flow control. As soon as the window size is increased, the TCP connection in the least favorable positions suffers from the traffic generated by the other two TCP sources, since priority is given to in-transit worms and, with deflection, nodes closer to the destination suffer from the traffic coming from farther nodes. The average throughput for TCP connections now ranges from 0.6 when no background is present to a limit of about 0.3 when the network is overloaded. This is an intermediate behavior with respect to the two sets of backpressure results: deflection is self-adaptive to the traffic situation, so that TCP connections tend to exploit well network resources. The problems related to interferences among fixed routes which must be faced both by wormhole with backpressure and by ATM, do not arise with deflection. Note that the reduced efficiency does not affect the goodput, which is still significantly larger with respect to the upper graph of Fig. 2. The better fairness obtained with a smaller window indicates that a control of the source rate is important to avoid congestion, as discussed in Ref. [5].

Results not presented here show that, in a network with 10 km links, the advantages of deflection control with respect to backpressure flow control are reduced, since increasing the network span implies that each deflection is more costly in terms of delay. Note that possible out-of-sequence packet receptions and varying delays are well tolerated by TCP.

Similarly to the backpressure case, the presence of background traffic breaks deterministic behaviors and increases fairness among TCP connections.

TCP connections are blocked or controlled at the network ingress when backpressure or deflection routing is considered; as a consequence, increasing the window size over a

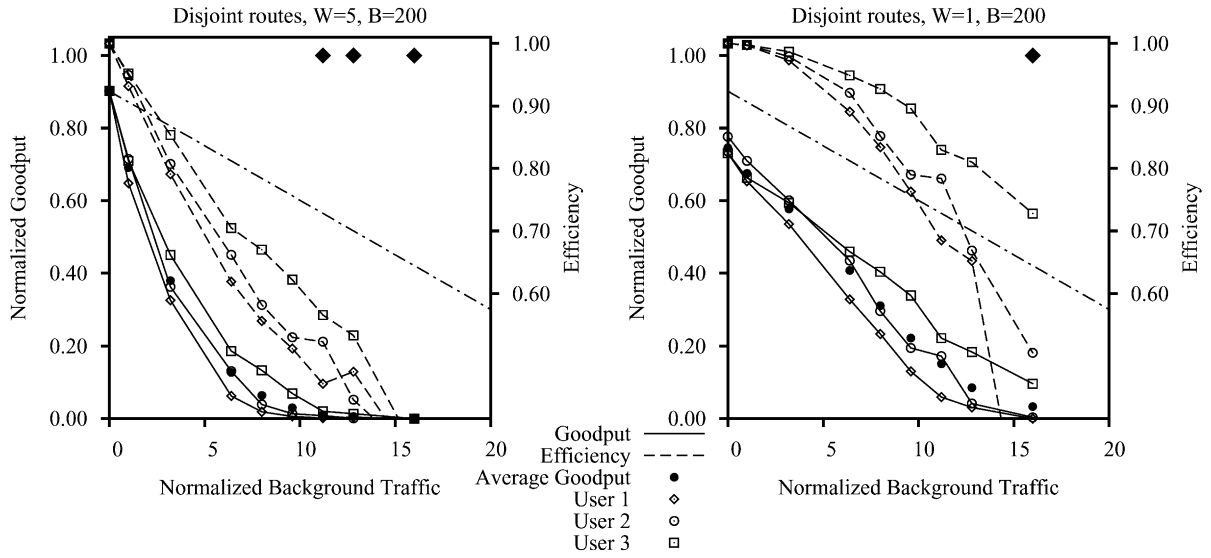


Fig. 4. Goodput and efficiency for ATM without traffic control and small buffers ($B = 200$ cells). Medium ($W = 5$) and small ($W = 1$) window sizes. Disjoint routes.

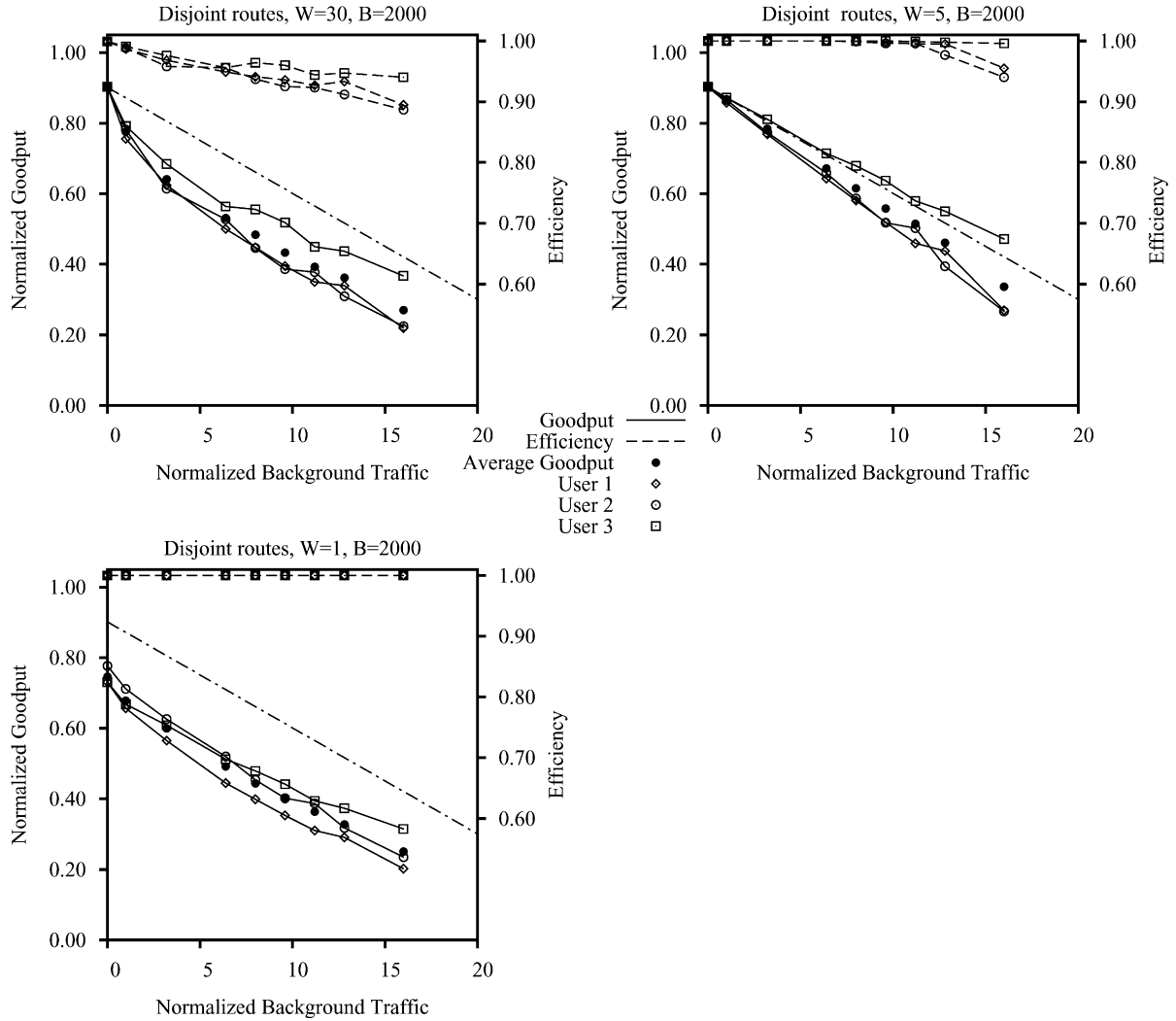


Fig. 5. Goodput and efficiency for ATM without traffic control and large buffers ($B = 2000$ cells). Large ($W = 30$), medium ($W = 5$), and small ($W = 1$) window sizes. Disjoint routes.

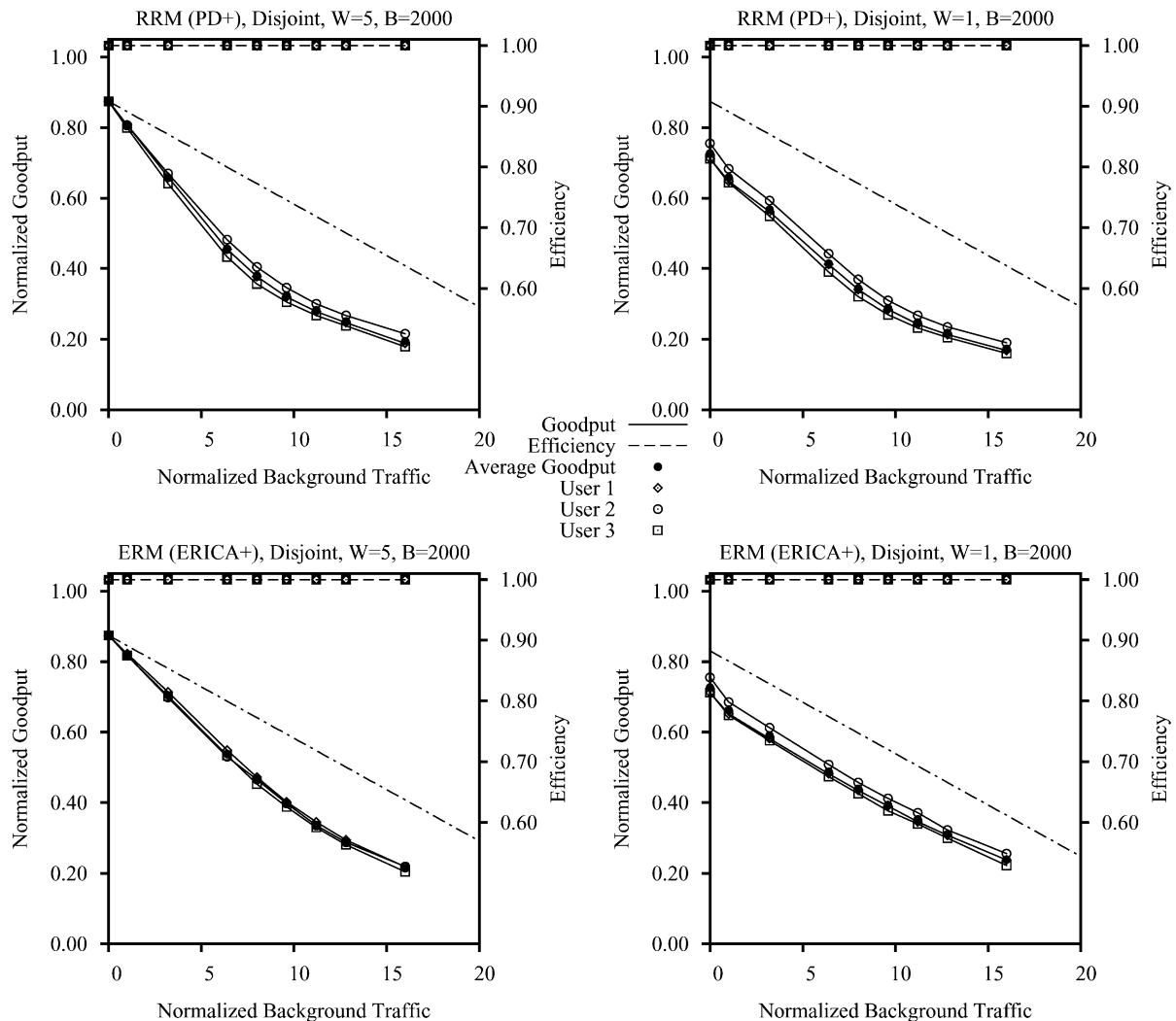


Fig. 6. Goodput and efficiency for ATM with ABR traffic controls and large buffers ($B = 2000$ cells). Medium ($W = 5$), and small ($W = 1$) window sizes. Disjoint routes.

certain limit has a negligible effect on goodput figures. A window size of five segments allows the TCP source to exploit the network resources; further increasing the window size does not influence the goodput performances, while efficiency figures are slightly worsened.

When presenting results for ATM networks, we first consider in Figs. 4 and 5 non-overlapping routes for TCP connections with UBR traffic control. Results show that an increase in the buffer size with respect to wormhole simulations must be introduced to obtain acceptable performance results. Indeed Fig. 4 shows that a buffer equal to 200 cells, which is approximately able to store one maximum size packet, leads to a cell loss rate which cannot be tolerated by TCP connections. The dot-dashed line is the maximum achievable average goodput, taking into account segmentation overheads and background traffic. The solid diamond appearing close to the upper edge of the graphs in correspondence with some simulation points means that one or more TCP connections were forced to close due to excessive

retransmissions. Note also the sharp decrease in efficiency for increasing background traffic.

Even worse performance was observed in the case of overlapping ATM routes, hence of interfering TCP connections.

Much better performance is instead achieved with 2000 cells buffers, as shown in Fig. 5. Note in Fig. 5 that increasing the window size over five segments worsens the performance of the TCP connections.

The adoption of a ABR traffic controls was observed to provide only a marginal improvement if a small buffer size is considered, while better performance is obtained with larger buffer sizes. However, Fig. 6 shows that the gains with respect to the uncontrolled (UBR) case are more in terms of fairness and of tolerance to large window sizes than in terms of achievable goodput. Probably, the ABR controls would have provided better performance with more sophisticated buffer management policies, as previously noted, but this would further add complexity in the comparison with wormhole routing.

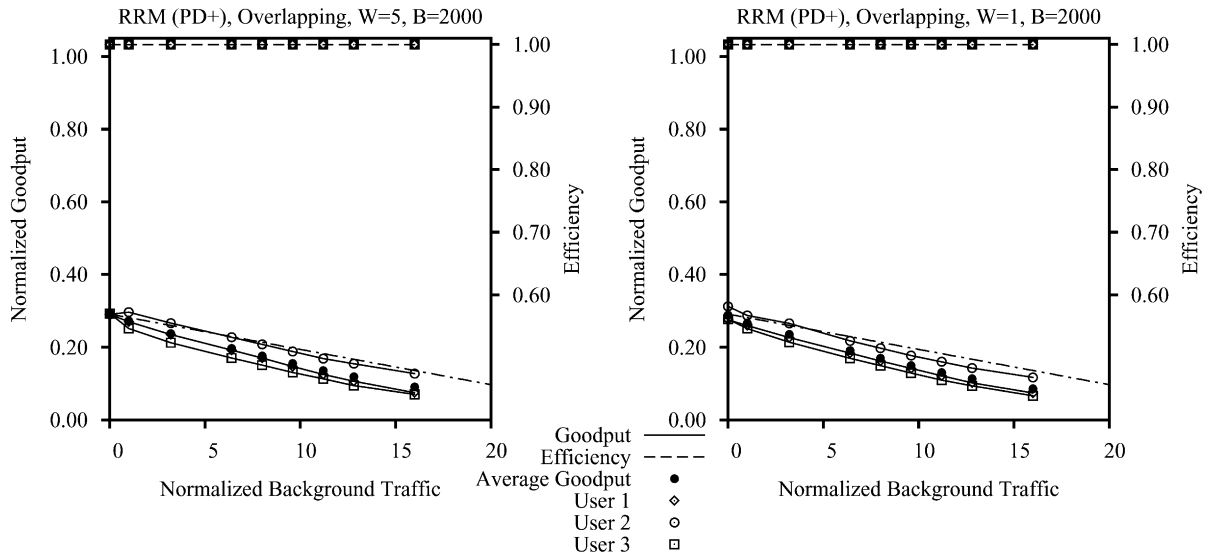


Fig. 7. Goodput and efficiency for ATM with RRM ABR traffic controls and large buffers ($B = 2000$ cells). Medium ($W = 5$) and small ($W = 1$) window sizes. Overlapping routes.

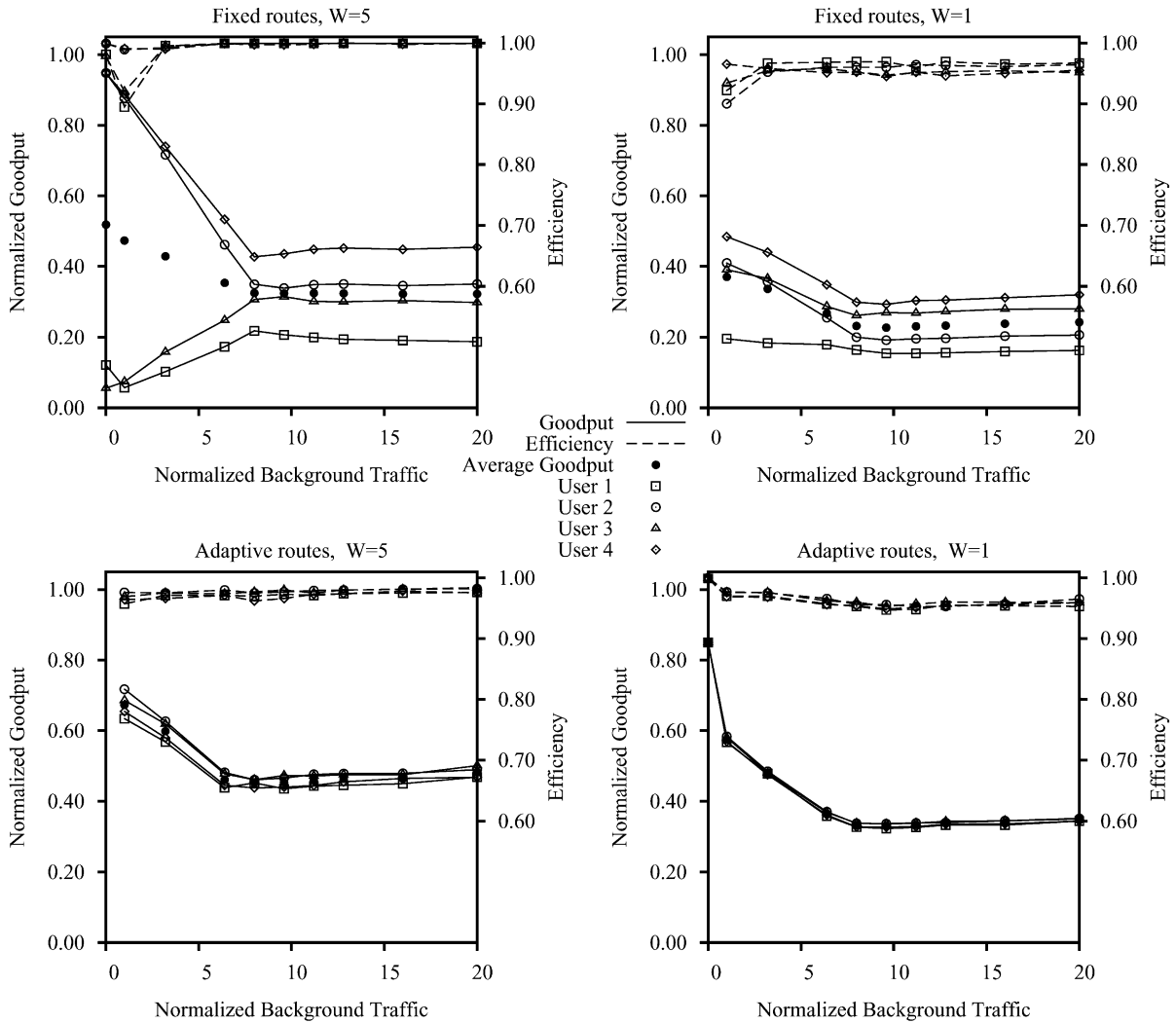


Fig. 8. Goodput and efficiency for wormhole routing with backpressure flow control, with overlapping fixed routes and adaptive routes. Medium ($W = 5$) and small ($W = 1$) window sizes. Linear configuration.

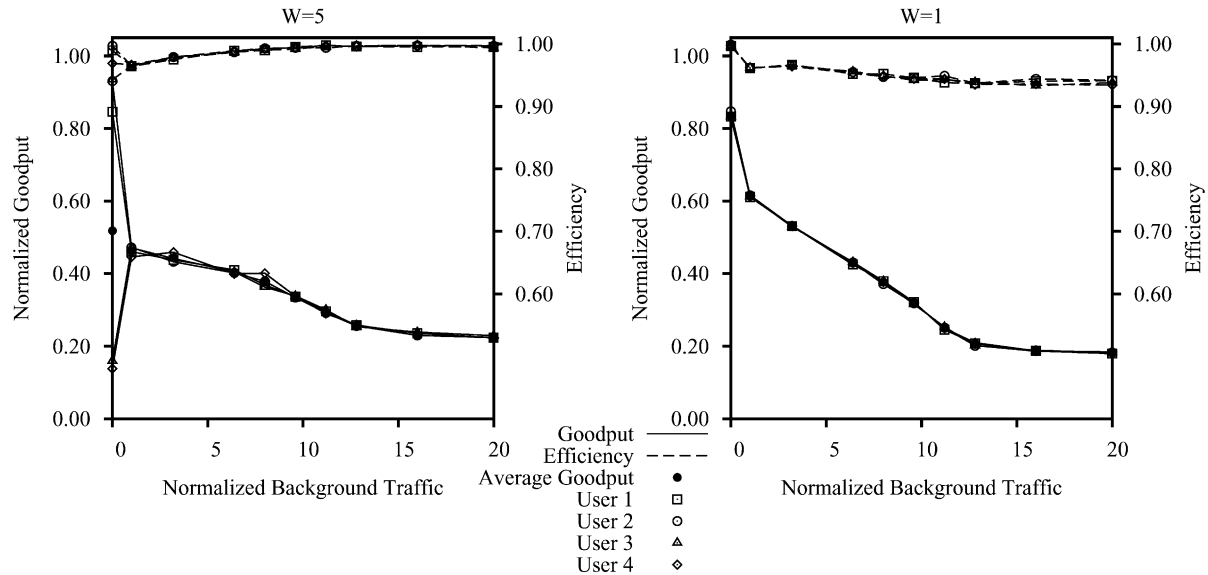


Fig. 9. Goodput and efficiency for wormhole routing with deflection flow control. Medium ($W = 5$) and small ($W = 1$) window sizes. Linear configuration.

Acceptable results can be obtained with ABR control also when the paths of the TCP connections share a single link (see Fig. 7, and compare it with Figs. 2 and 3); in this situation, but without ABR control, connections were forced to close for all non-negligible intensities of the background load.

In general the ERM ABR control provides better fairness, but not necessarily larger goodput, than the simpler RRM ABR control.

3.2. Second traffic scenario

We now present results for the second traffic scenario, where four TCP connections, pairwise, interfere in a ring of the 16-nodes Manhattan topology. Symmetries along the ring should provide similar performance to the four TCP connections. We refer to this traffic pattern as ‘linear configuration’.

Fig. 8 refers to the backpressure case, considering the case of fixed overlapping routes, and the case of adaptive, minimum-distance, routes. The unfairness that can be observed in the former case (upper part of the figure) is because of the virtual channel mechanism. Two virtual channels are used on each link of the ring where TCP connections are routed, and the virtual channel mapping is such that one of the two virtual channels bears more traffic than the other. While the packets transmitted by user 1 and user 2 only use the first virtual channel, packets of user 3 use both virtual channels, while packets of user 4 only traverse the second virtual channel, thereby obtaining better performance.

Note that the presence of background traffic has a beneficial effect on fairness, since deterministic behaviors are broken, and a shaping effect is introduced into TCP traffic by interferences with the background traffic.

The unfairness disappears in the case of adaptive routes, where all TCP connections can be routed in either direction along the ring: results in the lower part of Fig. 8 show better fairness and larger goodput, thanks to the more uniform exploitation of network resources.

Fig. 9 depicts results for wormhole routing with deflections. Good fairness is observed, and goodputs are comparable with backpressure flow control. Unfairness is observed only for $W = 5$ in absence of background traffic: this is due to an insufficient input rate control, which permits that sources 1 and 2 grab most of the resources on the ring.

Fig. 10 shows results for ATM networks, with large buffers, and UBR, ABR RRM, and ABR ERM traffic control. With respect to wormhole routing, ATM in general permits better fairness at the cost of a reduction of the average goodput and of additional complexity (larger buffers, connection-oriented approach, and traffic control algorithms). UBR provides good performance in this simple (and ‘easy’) traffic configuration. ABR Explicit Rate (ERICA) shows better fairness.

3.3. The parking lot traffic scenario

The Parking Lot traffic scenario is designed to enhance congestion problems: a sequence of TCP sources progressively adds congestion on a linear path. Downstream sources must face a larger congestion.

Figs. 11 and 12 show that wormhole networks, both with backpressure and with deflection flow control, have unacceptable fairness problems. Particularly for small intensities of the background traffic, downstream sources are almost starved.

ATM behaves much better in this scenario, as it can be observed in Fig. 13. Note that in this more ‘difficult’ traffic scenario, the additional complexity of ABR pays off: the

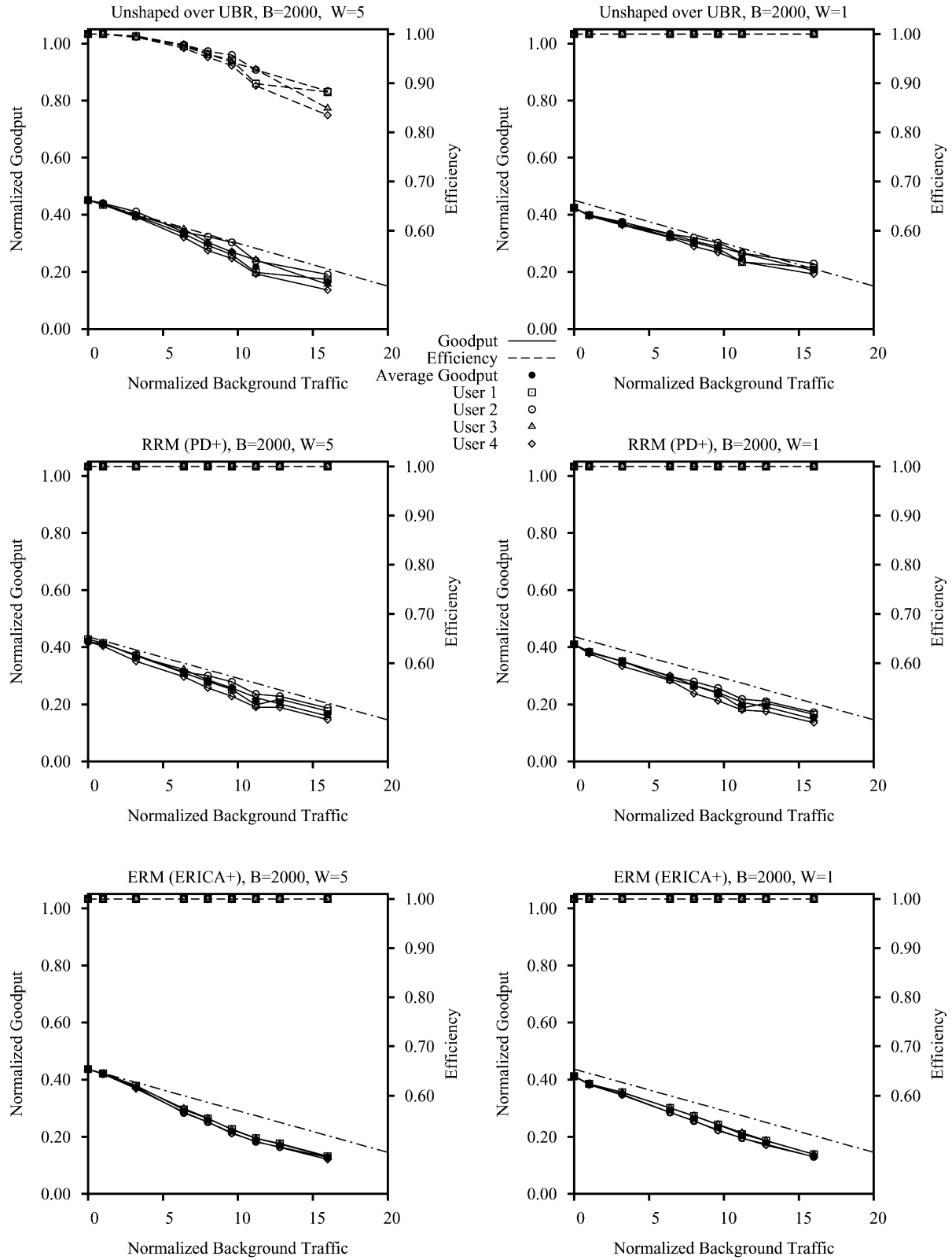


Fig. 10. Goodput and efficiency for ATM with UBR, ABR RRM and ABR ERM traffic controls, large buffers ($B = 2000$ cells). Medium ($W = 5$) and small ($W = 1$) window sizes. Overlapping routes in a linear configuration.

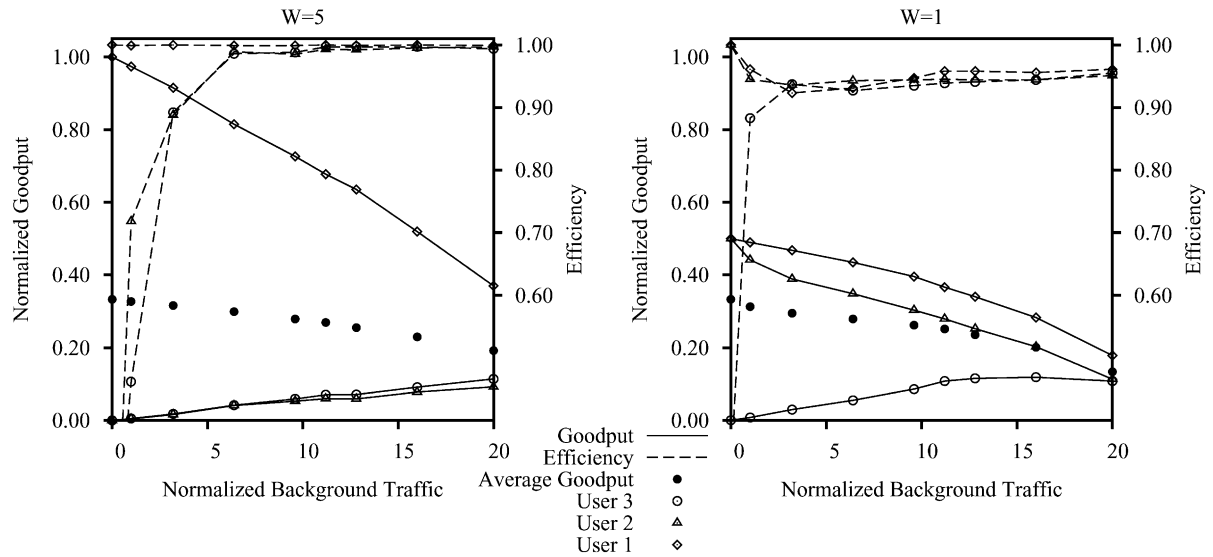


Fig. 11. Goodput and efficiency for wormhole routing with backpressure flow control. Medium ($W = 5$) and small ($W = 1$) window sizes. Overlapping routes in a parking lot configuration.

Explicit Rate approach combines large goodput with fairness for both window sizes.

4. Conclusions

In this paper we have presented a performance comparison between the ATM and the wormhole routing high-speed transport techniques under TCP overloaded connections.

Our simulation results show that ATM, in order to achieve average performance figures similar to those of wormhole routing, requires larger buffers and increased algorithmic complexity at switches. This probably also means larger costs for ATM. In general the additional

complexity of ATM pays off in terms of traffic control: better fairness and capability of dealing with congestion phenomena can be observed. The merits of wormhole routing are instead simplicity and larger overall throughputs.

Another interesting result is that deflection flow control provides a service to TCP connections which is well comparable with backpressure flow control in wormhole routing LANs, despite the problems of unbounded delays and out-of-order delivery of data units. This is mainly due to the fact that deflection is more adaptive to the network traffic, thereby permitting a better exploitation of network resources.

Two directions can be identified for the prosecution of

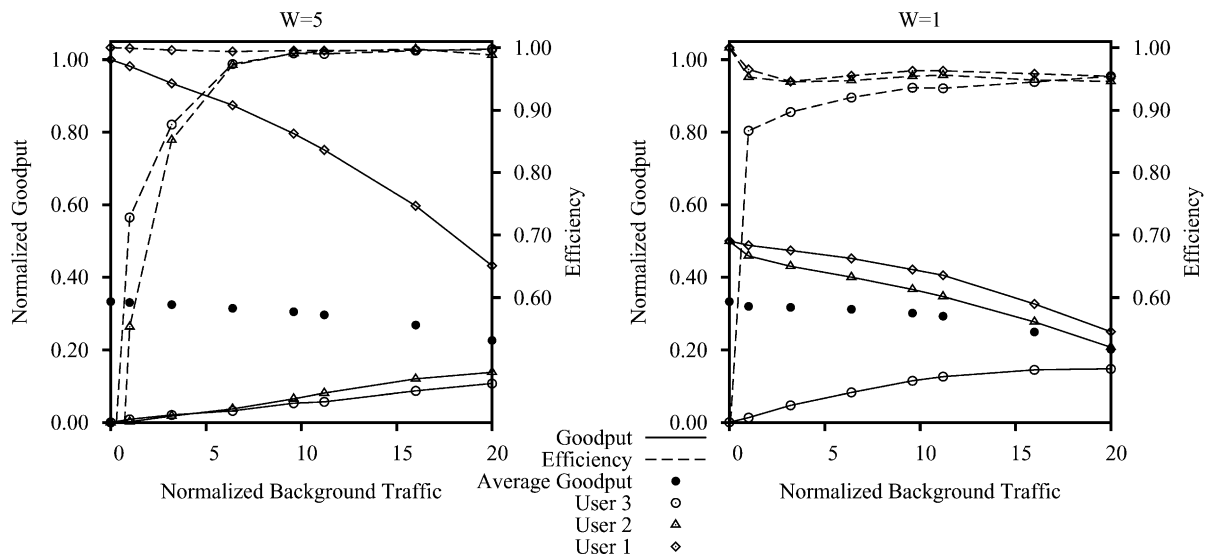


Fig. 12. Goodput and efficiency for wormhole routing with deflection flow control. Medium ($W = 5$) and small ($W = 1$) window sizes. Parking lot configuration.

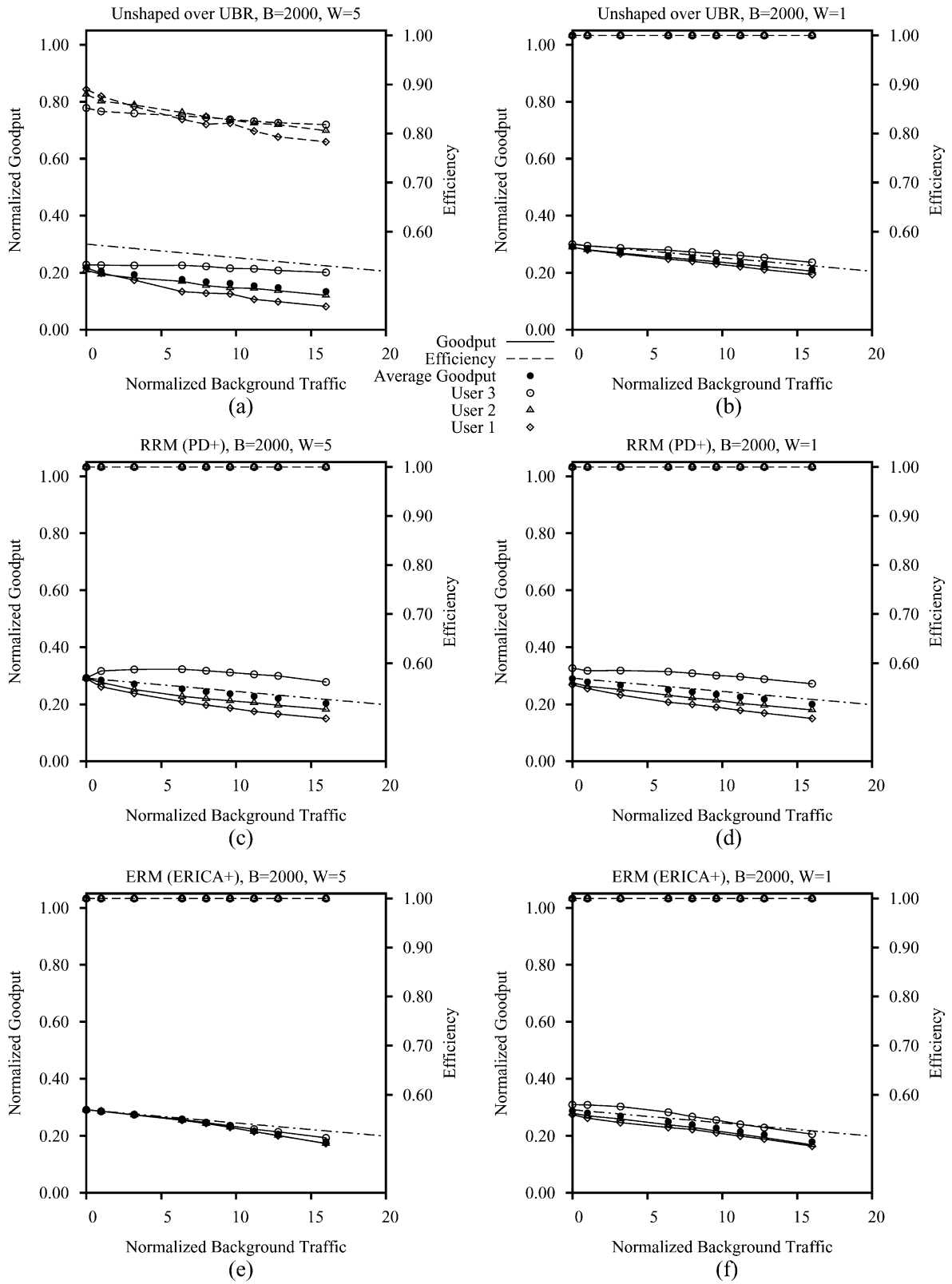


Fig. 13. Goodput and efficiency for ATM with UBR, ABR RRM and ABR ERM traffic controls, large buffers ($B = 2000$ cells). Medium ($W = 5$), and small ($W = 1$) window sizes. Overlapping routes in a parking lot configuration.

this work. On one side, different traffic scenario and different TCP source models (e.g. non-persistent sources) must be studied. On the other side, the wormhole routing and the ATM transport techniques should be compared with the IEEE 802.3u Fast Ethernet and the new IEEE 802.3z Gigabit Ethernet.

Acknowledgements

This work was supported in part by a research contract between Politecnico di Torino and CSELT, in part by the Italian Research Council (CNR) and in part by the Italian Ministry for University and Research.

References

- [1] P. Kermani, L. Kleinrock, Virtual cut-through: a new computer communication switching technique, *Computer Networks and ISDN Systems* 3 (3) (1979) 267–286.
- [2] Myricom, Inc., Myrinet in Brief, 1993.
- [3] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C.L. Seitz, J. Seizovic, W.-K. Su, Myrinet: a gigabit-per-second Local-Area Network, *IEEE Micro* 15 (1) (1995) 29–36.
- [4] W.J. Dally, C.L. Seitz, Deadlock-free message routing in multiprocessor interconnection networks, *IEEE Transactions on Computers* C-36 (5) (1987) 547–553.
- [5] E. Leonardi, F. Neri, M. Gerla, P. Palnati, Congestion control in asynchronous, high-speed wormhole routing networks, *Flow and Congestion Control*, G. Omidyar, G. Pujolle (Eds.), *IEEE Communications Magazine* 11 (1996) 58–69.
- [6] D. Kouvatsos (Ed.), *Performance Modeling and Evaluation of ATM Networks*, vol. 1, Chapman and Hall, London, 1996.
- [7] D. Kouvatsos (Ed.), *Performance Modeling and Evaluation of ATM Networks*, vol. 2, Chapman and Hall, London, 1996.
- [8] J. Postel, Transmission Control Protocol, RFC 793, September 1981.
- [9] R. Stevens, *TCP/IP Illustrated*, vols. I–III, Addison-Wesley, Reading, MA, 1994.
- [10] A. Ajmone Marsan, A. Bianco, R. Lo Cigno, M. Munafò, Four standard control theory approaches for the implementation of RRM ABR services, in: D. Kouvatsos (Ed.), *Performance Modeling and Evaluation of ATM Networks*, vol. 2, Chapman and Hall, London, 1997.
- [11] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, B. Vandalore, The ERICA switch algorithm for ABR traffic management in ATM networks, *IEEE/ACM Transactions on Networking* 8 (1) (2000) 87–98.
- [12] A. Bianco, Performance of the TCP protocol over ATM networks, *ICCCN'94*, San Francisco, CA, USA, September 1994.
- [13] M. Ajmone Marsan, A. Bianco, R. Lo Cigno, M. Munafò, Shaping TCP traffic in ATM networks, *IEEE ICT'95*, Bali, Indonesia, April 1995.
- [14] A. Romanow, S. Floyd, Dynamics of TCP traffic over ATM networks, *IEEE Journal on Selected Areas in Communications* SAC-13 (4) (1995) 633–641.
- [15] T. Lakshman, A. Neidhart, T. Ott, The drop from front strategy in TCP and in TCP over ATM, *IEEE INFOCOM'96*, San Francisco, CA, March 1996.
- [16] H. Li, K. Siu, H. Tzeng, A simulation study of TCP performance in ATM networks with ABR and UBR services, *IEEE INFOCOM'96*, San Francisco, CA, March 1996.
- [17] R. Goyal, R. Jain, S. Kalyanarama, S. Fahmy, S. Kim, UBR + : improving performance of TCP over ATM-UBR service, *IEEE ICC'97*, Montreal, Canada, June 1997.
- [18] R. Cole, D. Shur, C. Villamizar, I.P. over, ATM: A Framework Document, RFC 1932, April 1996.
- [19] P. Newnam, IP switching: ATM under IP, *INFOCOM'96*, San Francisco, CA.
- [20] V. Jacobson, Congestion avoidance and control, *ACM SIGCOMM'88*, Stanford, CA, USA, August 1988.
- [21] V. Jacobson, Berkeley TCP evolution from 4.3-tahoe to 4.3-reno, Eighteenth IETF, Vancouver, BC, Canada, August 1990.
- [22] V. Jacobson, R. Braden, D. Borman, TCP Extensions for High Performance, RFC 1323, May 1992.
- [23] M. Mathis, J. Mahdavi, S. Floyd, A. Romanov, TCP Selective, Acknowledgement Options, RFC 2018, October 1996.
- [24] D. Clark, Window and Acknowledgement Strategy in TCP, RFC 813, January 1982.
- [25] M. Ajmone Marsan, A. Bianco, T.V. Do, L. Jereb, R. Lo Cigno, M. Munafò, ATM simulation with CLASS, *Performance Evaluation* 24 (1995) 137–159.
- [26] ATM Forum AF-TM-0056.000, ATM Forum Traffic Management Specification, Version 4.0, April 1996.
- [27] R. Bagrodia, K.M. Chandy, J. Misra, A message-based approach to discrete-event simulation, *IEEE Transactions on Software Engineering* 13 (6) (1987) 664–665.
- [28] L.M. Ni, P.K. McKinley, A survey of wormhole routing techniques in direct networks, *IEEE Computer* 26 (2) (1993) 62–76.
- [29] R.C. Holt, Some deadlock properties of computer systems, *ACM Computing Surveys* 4 (3) (1972) 178–196.
- [30] K.D. Günther, Prevention of deadlocks in packet-switched data transport systems, *IEEE Transactions on Communications* COM-29 (4) (1981) 512–524.
- [31] J. Blazewicz, J. Brzezinski, G. Gambosi, Time-stamp approach to store-and-forward deadlock prevention, *IEEE Transactions on Communications* COM-35 (5) (1987) 490–495.
- [32] M.D. Schroeder, A.D. Birrell, M. Burrows, H. Murray, R.M. Needham, T.L. Rodeheffer, E.H. Satterthwaite, C.P. Thacker, Autonet: a high-speed self-configuring local area network using point-to-point links, *IEEE Journal on Selected Areas in Communications* SAC-9 (8) (1991) 1318–1335.
- [33] M. Gerla, L. Kleinrock, Flow control: a comparative survey, *IEEE Transactions on Communications* COM-28 (4) (1980) 553–574.
- [34] I.S. Gopal, Prevention of store-and-forward deadlock in computer networks, *IEEE Transactions on Communications* COM-33 (12) (1985) 1258–1264.
- [35] P.M. Merlin, P.J. Schweitzer, Deadlock avoidance in store-and-forward networks — I: store-and-forward dead-lock, *IEEE Transactions on Communications* COM-28 (3) (1980) 345–354.
- [36] P.M. Merlin, P.J. Schweitzer, Deadlock avoidance in store-and-forward networks — II: other deadlock types, *IEEE Transactions on Communications* COM-28 (3) (1980) 355–360.
- [37] P. Palnati, M. Gerla, E. Leonardi, Deadlock-free routing in an optical interconnect for high-speed wormhole routing networks, *International Conference on Parallel and Distributed Systems (ICPDS)*, June 1996, pp. 256–264.
- [38] N. Maxemchuk, Routing in the manhattan street network, *IEEE Transactions on Communications* COM-35 (5) (1987) 503–512.
- [39] N. Maxemchuk, Comparison of deflection and store-and-forward techniques in the Manhattan street and shuffle-exchange networks, *IEEE INFOCOM'89*, vol. 3, 1989, pp. 800–809.
- [40] F. Borgonovo, E. Cadorin, Locally optimal deflection routing in the bidirectional manhattan network, *IEEE INFOCOM'90*, June 1990, pp. 458–464.
- [41] C. Baransel, W. Dobosiewicz, P. Gburzynski, Routing in multihop packet switching networks: Gb/s challenge, *IEEE Network* 9 (3) (1995) 38–61.