DIMACS Technical Report 99-04
January 1999

Combinatorial Design of Multi-Ring Networks with Combined Routing and Flow Control

by

Avishai Wool [1]
Bell Laboratories, Lucent Technologies,
700 Mountain Ave., Murray Hill, NJ 07974.
E-mail: yash@research.bell-labs.com

Bülent Yener [2]
Bell Laboratories, Lucent Technologies,
700 Mountain Ave., Murray Hill, NJ 07974
E-mail: yener@research.bell-labs.com

[1]Permanent Member
[2]Permanent Member

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs, Bellcore, and Bell Labs.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

In this paper we present a novel design technique, and combined routing and flow control algorithms, for congestion-free packet switched networks.

The design is based on the construction of multiple virtual rings, under the constraint that the path between any two nodes is either confined to a single ring, or traverses exactly two rings (passing through a single bridge node).

Our best designs are constructed by using *Finite Generalized Quadrangles* of combinatorial design theory, together with a *scaling* algorithm for realizing networks of arbitrary size. The target topology is obtained by taking the edge union of the multiple virtual rings.

Capitalizing on the underlying topological properties, we design routing, flow and access control protocols. We prove that the proposed network architecture, coupled with our protocols, ensures that (i) no loss due to congestion occurs inside a network, under arbitrary traffic patterns; (ii) all the packets reach their destinations within bounded time; and (iii) the bandwidth is allocated fairly and no host is starved.

Moreover, we achieve these desirable properties much more efficiently than earlier proposals. Our best designs have a maximum route length of $O(N^{1/3})$ and require $O(N^{1/3})$ ports to be used at each node for an N-node network. The designs attain these bounds while manifesting a high degree of redundancy, with multiple disjoint paths between all pairs of nodes. This significantly improves upon the designs of [YOY94, YOY97], where both the route length and number of ports are $\Omega(\sqrt{N})$, and only a single path exists between every pair of nodes.

We also provide a theoretical bound for the impact of fairness and congestion-freeness on network utilization. We used this design technique to generate network designs of various sizes. We then implemented our protocols and verified their performance by simulations.

1 Introduction

The ever-increasing growth of data traffic and its bandwidth demand necessitates careful design and planning of the infrastructures of next generation networks. This paper presents a novel network design methodology and shows how routing, flow and access control protocols can be integrated to capitalize on the design properties.

Due to the complexity of the problem, traditional approaches to network design either face computationally hard problems, or use heuristic methods with approximate answers. In contrast, our proposed design methodology is based on combinatorial design theory. The combinatorial properties of the designs themselves translate directly into bounds on the maximum route length and on the survivability of network, bounds which are independent of the traffic characteristics. Ensuring traffic-independent properties is particularly important for data networks where traffic is bursty and unpredictable.

In the first stage of our design technique, we use *Generalized Quadrangles* (GQs), to obtain a network of multiple rings that has the *one-bridge* property: the path between any two nodes is either confined to a single ring, or traverses exactly two rings (passing through a single bridge node).

Our design technique can actually use any subgraph (e.g., a bus or a star) as a basic building block. We focus on ring networks for several reasons. The ring topology is well studied, and has been used in LAN and MAN environments. Its attractive symmetry and cyclic structure allow for simple, decentralized, fair, and congestion-free control protocols. Since a ring is a minimal biconnected graph it also provides survivability with minimal cost. For example, a multi-ring topology is the backbone topology of choice in the Synchronous Optical Network (SONET) standard [ANS88]. Recent surveys on other uses of multi-ring networks are [DH98, LSBW98].

The next stage in our technique is to design control protocols that capitalize on the combinatorial properties of our multi-ring networks. We do this in the framework of virtual slotted ring emulation [LOY96, BOY97]. Within this framework we propose a combined routing, flow and access control protocol that relies on the one-bridge property. The proposed network architecture, coupled with our protocol, ensures that (i) the network is congestion-free, under arbitrary traffic patterns; (ii) all the packets reach their destinations within bounded time; and (iii) the bandwidth is allocated fairly and no host is starved.

Moreover, our method guarantees convergence and is throughput scalable (adding new nodes or links will not decrease the throughput). Current network designs do not ensure both these properties simultaneously. Networks with a simple linear topology, such as a bus or a ring [FN69, MB76, BCK+83], are not throughput scalable. In contrast, networks with an arbitrary topology are throughput scalable but they are typically not congestion-free.

Combinatorial design theory was first applied to multi-ring network designs in [YOY94, YOY97]. The authors showed how to use *Balanced Incomplete Block Designs* (BIBDs) to obtain congestion-free networks with scalable throughput. The techniques resulted in networks in which the maximum route length and the maximum degree (number of rings a node belongs to) are both bounded by $O(\sqrt{N})$, where N is the number of nodes in the network. These bounds are similar to those of earlier approaches to multi-ring network design (e.g., chordal rings [RGA85, RS86] or ring-connected-ring [DP91]), with the additional property of congestion-free routing. In this paper we continue to guarantee congestion-free routing, yet we improve both the bound on the maximum route length and the bound on the node degree to $O(N^{1/3})$.

In addition to the above-mentioned properties, our networks also enjoy several other attractive characteristics. We use bounded (fixed sized) buffers within the network. Our GQ-based networks have a high level of path redundancy which translates to high survivability in the face of link and node failures, and allows easy load balancing among rings. And our networks require only $O(N^{4/3})$ links in total, which is cheaper than the $O(N^{3/2})$ links of [YOY94] and only slightly more expensive than the minimal $N-1$ links required for connectivity.

To evaluate the performance of our networks beyond the analytical worst-case bounds, we conducted an extensive simulation-based study. The simulation included a complete implementation of our protocols. Our results confirm that the networks do indeed scale up well and may offer substantial gains.

The rest of this paper is organized as follows. Section 2 explains the network design model and introduces the generalized quadrangle designs. Section 3 contains a description of the basic slotted ring operation. In Sections 4 and 5 we describe our access control and flow control protocols. In Section 6 we discuss the properties of our networks. In Section 7 we show how to design networks of arbitrary size. In Section 8 we describe the results of a simulation-based performance evaluation of our protocols, and we conclude in Section 9.

2 Combinatorial Construction of Rings

The basic design criterion in all our multiple ring network designs is that they should obey the *one-bridge* property. By this we mean that a packet would need to cross at most one ring-to-ring bridge along its path from any node to any other node in the network. We shall see that this property lets us design networks with small rings (giving low propagation delay) and low degree—and at the same time admits very efficient congestion-free flow-control protocols. Formally we use the following definition.

Definition 2.1 *Let $\mathcal{R} = R_1, \ldots, R_M$ denote the rings of a multi-ring network. Then $\mathcal{R}$ is*

said to have the one-bridge *property if for every two nodes x, y one of the following conditions holds:*

1. *There exists a ring R_i such that $x \in R_i \ni y$. We say that x and y are* neighbors *on ring R_i.*

2. *There exist two rings R_i, R_j and a node z such that $x \in R_i \ni z \in R_j \ni y$. In this case we call z the* bridge node*, or simply the bridge.*

Clearly block-designs such as those used in [YOY94] have the one-bridge property in a trivial way: every two nodes are neighbors on some ring, so condition 1 of Definition 2.1 always holds. In this paper we focus on designs in which many (indeed, the vast majority) of the node pairs are *not* neighbors on any ring.

2.1 A Tradeoff between Ring Size and Node Degree

The one-bridge property constrains the characteristics of the achievable designs. In particular there is a trade-off between the ring size n and the node degree d on an N-node network. However the tradeoff imposed by our one-bridge property is much better than that of [YOY94]: their designs had $nd = \Omega(N)$ so n and d could achieve at most $O(\sqrt{N})$ simultaneously. In contrast, we have $nd = \Omega(\sqrt{N})$ so we can hope to achieve n and d of $O(N^{1/4})$ simultaneously.

Theorem 2.2 *Consider a multi-ring network over N nodes that has the one-bridge property. Let n denote the size of the largest ring and let d denote the maximal node degree. Then*

$$nd = \Omega(\sqrt{N}).$$

Proof: Consider some node u. This u has at most $d(n-1)$ distinct nodes as neighbors. Each of these neighbors belongs to at most $d-1$ rings which u does not belong to, so u has at most $d(n-1)(d-1)(n-1)$ distinct neighbors-of-neighbors. By the one-bridge property, every node $v \neq u$ is either a neighbor or a neighbor-of-a-neighbor of u. Hence

$$N \leq 1 + d(n-1) + d(n-1)(d-1)(n-1) = (nd)^2(1 + o(1)). \qquad \blacksquare$$

2.2 Generalized Quadrangles

Generalized Quadrangles (GQs) are a rich class of combinatorial designs that enjoy the one-bridge property of Definition 2.1. A succinct survey of these combinatorial objects can be found in [CD96, ch. IV.21]. An in-depth mathematical treatment of GQs can be found in the monograph [PT84]. Even a brief introduction to the theory of GQs would exceed the scope

{1,4,5}	{2,4,6}	{3,4,7}	{5,10,15}	{6,11,13}
{1,8,9}	{2,8,10}	{3,8,11}	{5,11,14}	{7,9,14}
{1,12,13}	{2,12,14}	{3,12,15}	{6,9,15}	{7,10,13}

Table 1: The $W(2)$ generalized quadrangle design, with $N = 15$ nodes of degree $d = 3$, and $M = 15$ rings of size $n = 3$.

of this paper, so we limit ourselves here to the very basic properties of GQs, with emphasis on how they fit into our methodology of network design.

Definition 2.3 [CD96, IV.21.1] *A* finite generalized quadrangle (GQ) *is a collection $\mathcal{R}$ of sets (rings) over a universe U of nodes satisfying the following:*

1. *each node belongs to $t + 1$ rings, and two distinct nodes belong to at most one ring together;*
2. *each ring contains $s+1$ nodes, and two distinct rings have at most one node in common;*
3. *if $x \in U$ is a node and $R \in \mathcal{R}$ is a ring such that $x \notin R$ then there exist a unique $y \in U$ and $S \in \mathcal{R}$ such that $x \in S \ni y \in R$.*

A GQ *with parameters s, t is said to be of* order (s, t).

The next proposition shows that not only do the GQs have the one-bridge property, they have additional symmetry properties that are useful for our routing protocols.

Proposition 2.4 *Let x, y be nodes in a* GQ *of order (s, t). If x and y are not neighbors then there exist $t + 1$ distinct pairs of rings R_i^x, R_i^y such that $x \in R_i^x$, $y \in R_i^y$, and $|R_i^x \cap R_i^y| = 1$ for all $i = 1, \ldots, t + 1$.*

Proof: By definition, x belongs to $t + 1$ rings, which we denote by R_i^x for $i = 1, \ldots, t + 1$. The node y does not belong to any of these R_i^x rings since x, y are not neighbors. So by condition 3 of Definition 2.3, for every such R_i^x there exists a unique R_i^y which contains a single bridge node z_i such that $x \in R_i^x \ni z_i \in R_i^x \ni y$. ∎

Note that Proposition 2.4 proves much more than the one-bridge property. It in fact shows that if x needs to send a packet to y and they are not neighbors, then x can select *any* of the $t + 1$ rings it is a member of and place the packet on this ring. The proposition shows that on each of x's rings there exists a bridge that also belongs to one of y's rings and is capable of bridging the packet to its destination. Thus using a GQ as the underlying design of the network gives us a great deal of flexibility in our routing protocol, which we collectively call "*self-routing*": The source x does not really need to know how to route a packet to a destination y that is not a neighbor—the packet can be placed on any arbitrary ring, and it will arrive at y.

Proposition 2.5 (Salient parameters of a GQ**)** [CD96, IV.21.3] *A* GQ *of order (s, t) has*

Nodes (N)	Rings (M)	degree (d)	ring size (n)	neighbors
15	15	3	3	6
27	45	5	3	10
40	40	4	4	12
45	27	3	5	12
64	96	6	4	18
85	85	5	5	20
96	64	4	6	20
112	280	10	4	30
125	175	7	5	28
156	156	6	6	30
165	297	9	5	36
175	125	5	7	30
280	112	4	10	36
297	165	5	9	40
325	1105	17	5	68
343	441	9	7	54
400	400	8	8	56

Table 2: The parameters of known GQ constructions, with up to $N = 400$ nodes.

- *a total of* $N = (s+1)(st+1)$ *nodes;*
- *a total of* $M = (t+1)(st+1)$ *rings;*
- *rings of size* $n = s+1$*;*
- *node degree of* $d = t+1$*;*
- *each node has* $ts + s$ *nodes as neighbors.*

A trivial example of a GQ is a grid. Arrange the N nodes in a $n \times n$ grid. Each row and each column in the grid constitutes a ring, giving $M = 2n$ rings in total. It is easy to see that this is a GQ of order $(n-1, 1)$. For such grid GQs Theorem 2.2 is tight up to a constant factor of 2.

We are more interested in other, non-trivial, GQ constructions, in which the ring size grows more slowly than $\sqrt{N}$. We see from Proposition 2.5 that GQs have $n^2 d = \Omega(N)$. Thus, balancing the constraints on n and d, we can hope to have $n = d = O(N^{1/3})$. It so happens that infinite families of such constructions are known to exist. For instance, there exists a GQ of order (q, q) (so called the $W(q)$ design) for every q which is a power of a prime number.

As an example, in Table 1 we show the complete $W(2)$ design, which is the smallest non-trivial GQ and has an order of (2,2). Table 2 lists the salient parameters of the known

GQ constructions with up to 400 nodes. We refer the reader to [CD96, ch. IV.21] and to the monograph [PT84] for the precise details of each construction.[1]

3 Virtual Slotted Ring Operation

There are several possible ring emulation methods we can use, as discussed in [BOY97] in the context of multicasting. In this section we begin to describe the details of the slotted ring emulation we suggest. We are using the *time-driven priority* technique [LOY96] to slot the link bandwidth in time.

3.1 Preliminaries

Our basic assumptions for this emulation are as follows.

1. Each ring has a unique ID R_i. Each node u has a virtual ID (VID) which is the set of its virtual ring IDs (e.g., if a node belongs to rings R_1, R_3, R_7, then $VID(u) = \{R_1, R_3, R_7\}$).

2. Each node u knows the *physical* address v and the VID of each of its ring-neighbors on every ring it belongs to. Each node also knows the network design structure (i.e., which nodes belong to each ring).

3. Each packet header carries the physical address of its destination. Using the information in (1) and (2), each node u can perform a routing-table-check function for a given destination v to determine the virtual ID of destination v.

4. A packet $P(v)$ sent from source u to destination v is called $LOCAL$ if $VID(u) \cap VID(v) \neq \emptyset$ (i.e., u and v are neighbors on some ring). Otherwise $P(v)$ it is called $REMOTE$.

5. For each $REMOTE$ packet $P(v)$ originating at u, u is able to select a ring R_i for $P(v)$ which contains the *bridge* node x guaranteed by the one-bridge property. Namely, $R_i \in VID(u) \cap VID(x)$ and also $VID(x) \cap VID(v) \neq \emptyset$.

6. Precise external timing information is available to each node in the network (e.g., using the global positioning system (GPS) [Kap96, Dan98]).

Notation: For simplicity we assume that the network is completely symmetric. In particular we assume that all the rings have the same number of nodes, denoted by n, and all the nodes

[1] We have also constructed some non-GQ designs with the one-bridge property. We omit the details for lack of space.

have the same *degree* d (i.e., each node belongs to exactly d rings). Note that the GQ-based constructions we described in Section 2 indeed enjoy this symmetry.

3.2 Network Timing and Bandwidth Allocation

The network's timing is based on a global referencing structure such as GPS. Thus, each node in the network receives clock ticks from a GPS receiver and uses them to keep its internal clock synchronized. The time between two successive ticks is called a *Time Slot* (TS). A typical duration for a TS is 125 μsec or 8KHz.

We assume that the slots rotate clockwise at each clock tick and the time interval between two consecutive ticks is long enough to insert or remove a packet. Each slot has a 1-bit status field marking the slot as being either *FULL* (i.e., carrying a packet) or *EMPTY*.

We denote the time it takes for a slot to rotate all the way around the ring by D, the link data rate by C, and the number of slots (packets) simultaneously on a ring by p. So for packets of size s the link data rate must obey the inequality $C \geq \frac{ps}{D}$.

We assume that one of the nodes (say the one with minimum ID number) in each ring is assigned as the *virtual ring bandwidth manager*. The manager node is the origin of the ring and the slots are numbered from 1 to p according to the time that the origin "sees" them.

4 Access Control

4.1 Quota-based Schemes

The purpose of our access control is to ensure fairness among different nodes. For this we introduce a quota which we denote by k, and allow a node u to transmit up to k packets on each ring R_i that it belongs to, during any cycle of duration D. Naturally, we set

$$k = \lfloor p/n \rfloor.$$

A trivial way to implement this access control is to statically reserve k slots for each node on each ring it belongs to, and to allow a node u to insert packets only to these designated slots. This approach is rather inflexible, and may introduce unnecessary delays, e.g., when a node has a burst of k packets to send, and it sees *EMPTY* slots, yet these slots belong to some other node $v \neq u$.

Instead we suggest a more flexible implementation which is based on quota counters. For each ring it belongs to a node u holds a quota-counter, which is replenished to k each cycle. u can send packets as long as the counter is positive, and the counter is decremented for every packet sent. However we need to be careful about how the quotas are replenished. A naive

implementation of a quota-based access control could lead to unfairness, as the following example shows.

Example 4.1 (Unfair quota implementation) Suppose that the nodes on R_i replenish their quotas to k at times $t = 0, D, 2D, \ldots$ (i.e., at the beginning of each cycle). Then it may be possible for node u to send k packets on the last k slots it sees in the cycle which ends at time t, and k more on the first k slots it sees in the next cycle, without violating its quota in either cycle. However note that between time $t - k$ and $t + k$ node u used $2k$ slots. From the point of view of a node v downstream from u, all these slots may appear during a single cycle, which in turn may cause v not to be able to send any packets in that cycle.

To avoid this phenomenon, we need to strengthen what we require of the nodes. This is formalized by the following invariant which we require the protocol to obey.

Invariant 4.2 *Every node u may send at most k packets within* any *consecutive period of D time (p clock ticks).*

The next technical definition is useful in the proof of Theorem 4.6.

Definition 4.3 *Let the* window $W_t(u) = (a_1, \ldots, a_p)$ *be a p-tuple containing the numbers of the slots that node u sees between time t and $t + D$. Thus $W_t(u)$ is a cyclic shift of $(1, \ldots, p)$ (see Figure 1).*

Fact 4.4 *Let $\Delta(v, u)$ be the* time difference *between node v and u, such that a slot seen by node v at time t is seen by u at time $t + \Delta(v, u)$. Then $W_t(v) = W_{t+\Delta(v,u)}(u)$ and $W_t(u) = W_{t-\Delta(v,u)}(v)$.*

Node	Delay	Slots seen																	
o		1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
v	$\Delta(o, v) = 2$	8	9	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7
u	$\Delta(v, u) = 3$	5	6	7	8	9	1	2	3	4	5	6	7	8	9	1	2	3	4

Figure 1: The slots of one ring as seen by three hosts over 2 cycles, starting at time t. Thus $W_t(u) = (5, 6, 7, 8, 9, 1, 2, 3, 4)$.

We can equivalently state Invariant 4.2 using Definition 4.3 as follows.

Invariant 4.5 *For every mode u and time t, u can use at most k slots in the window $W_t(u)$.*

When the access control protocol obeys Invariant 4.5 we can prove the following theorem, which both shows that the protocol is fair and gives a bound on the access delay.

Theorem 4.6 *If node u has $k' \leq k$ packets to send at time t, then u will succeed in sending all of them by time $t + D$.*

Proof: We first observe that a packet cannot stay on a ring R_i more that D time. If the packet is *LOCAL* then it will be removed by its destination within at most D time. If it is *REMOTE* then it will reach the bridge node on R_i within at most D time and will be

removed.[2]

Consider the window $W_t(u)$ which contains the sequence of p slots that u sees at times starting at time t. Assume w.l.o.g. that $W_t(u) = (a, a+1, \cdots, a+p-1) \bmod p$. If u cannot use slot $a+j$ it is because some other node v placed a packet in slot $a+j$ earlier. Since a packet cannot occupy a slot more than D time, v must have placed the packet in slot $a+j$ during the window $W_{t-\Delta(vu)}(v)$. Thus each slot that u sees as used by v in $W_t(u)$ corresponds to a packet that was placed by v within the window $W_{t-\Delta(vu)}(v)$. Since v obeys Invariant 4.5, v used at most k slots in window $W_{t-\Delta(vu)}(v)$. So by Fact 4.4 there are at most k such slots in $W_t(u)$. Since no packet can occupy a slot more than D time u will never see a slot marked by itself. Thus, at most $(n-1)k$ slots are will be occupied when u sees them in $W_t(u)$. Since $p \geq nk$ there will be at least $p-(n-1)k \geq k$ empty slots for u to use in the window $W_t(u)$, and u will be able to send its k' packets. ∎

Remark: Theorem 4.6 does not distinguish between different types of traffic. The k packets per cycle that a node is allowed to send contain both new packets (that the node needs to inject into the network) and bridged packets (which were originated by some other node and are already in the network). So Theorem 4.6 in itself is not yet sufficient to provide bandwidth or delay guarantees to the nodes; these would also depend on the flow control protocols used for the different traffic types.

4.2 Implementing the Scheme

Invariant 4.5 can maintained by a *window-based access* scheme. The scheme is different from standard window-based flow control schemes (cf. [Tan96]) since it is performed only at the senders, without any explicit acknowledgments. In this method, each node maintains the following information: (1) The window size $ws(t)$, (2) a δ-list of timers, and (3) the last-sent-time. The variable $ws(t)$ is simply a counter indicating the number of slots that can be used by node u at time t. Initially $ws(0)$ is set to k. After sending a packet, the counter $ws(t+1)$ is set to $ws(t)-1$ and a time stamp of $t+D$ is assigned to that packet. The time stamps are organized into a δ-list, which is a linked list of integers indicating when the timer of the packet just sent will expire, with respect to the deadline of the previous packet sent. The head of the linked list contains the earliest deadline, which is decremented by one at each slot-tick. When the value becomes zero at time t', the list is shrunk and the current window is incremented (i.e., $ws(t') \leftarrow ws(t'-1)+1$).

[2] We prove in Theorem 5.4 that there is no buffer overflow at the bridge. However for the purpose of access control this is not crucial, as long as a the packet is removed from the ring by the bridge.

4.3 Link Utilization with Access Control

Since we are using an access control scheme to guarantee fairness among the nodes, we are also bounding the maximal link utilization that can be achieved. In this subsection we quantify this bound. In particular we show that if the distribution of destinations on the ring is uniform then the utilization cannot exceed 50% when access control is used. Clearly if the requirements of *no-loss due to congestion* and *fairness* are relaxed (i.e., if we over-allocate the bandwidth based on statistical multiplexing), the utilization can be driven to a much higher value.

Consider a ring R_i of p slots operating for T cycles. We say that a packet that occupies a slot for one clock tick uses one *slot-tick*. Thus during a cycle of p clock ticks at most p^2 slot-ticks may be used, and during T cycles the total number of slot-ticks is p^2T.

The next two definition capture our notions of ring utilization and average path length, which we need for Proposition 4.9.

Definition 4.7 *For a packet z let $l(z)$ denote the* length *of the path that z traverses on the ring, i.e., the number of slot-ticks z uses. Then the* utilization *of the ring over T cycles is $Util = \left(\sum_z l(z)\right)/p^2T$, where the summation is over all the packets that are injected during these T cycles.*

Definition 4.8 *Let $\#packets$ denote the total number of packets injected onto the ring during T cycles. Then $E[PLen] = \left(\sum_z l(z)\right)/\#packets$.*

Proposition 4.9 *If the access control allows at most p packets to be injected onto the ring during a cycle, then*

$$Util \leq \frac{E[PLen]}{p}.$$

Proof: By plugging Definition 4.8 into Definition 4.7 we obtain that

$$Util = \frac{E[PLen] \times \#packets}{p^2T}.$$

If only p packets can be injected per cycle, we have that $\#packets \leq pT$ and we are done. ∎

Remarks:

- The proposition is a quantified version of the well known tradeoff between fairness and optimality (of the utilization, in our case) [Tan96]. Note that it holds for *any* fair access control scheme on a slotted uni-directional ring and is not an artifact of our method.
- If the distribution of destinations on the ring is uniform then $E[PLen] = p/2$, so Proposition 4.9 shows that $Util \leq 0.5$. A utilization close to 1 can only be achieved if

$E[PLen] \approx p$, i.e., each node sends packets only to its furthest possible destination on the ring.

- The proof ignores the "boundary" effects, namely, that initially the ring may not be empty, and that packets sent during cycle number T may reach their destinations only after the end of this cycle. It is easy to see that when T is large these effects are negligible.

The proof implies that no protocol can do better on a ring if the protocol ensures fairness and no-loss due to congestion. This is the first (best to our knowledge) provable, protocol-independent, theoretical bound on the effect of the fairness and no-congestion requirements on link utilization on ring architectures.

5 Flow Control

The purpose of our flow control is to ensure that the network is congestion-free. Clearly packets are not dropped while they are on a single ring, so *LOCAL* traffic is certainly congestion-free. Therefore we need to address *REMOTE* traffic, and specifically we need to ensure that packets are never dropped because the buffers at a bridge overflow. We are able to do this efficiently by capitalizing on the one-bridge property again—a *REMOTE* packet needs to cross exactly one bridge, so we need to be concerned with only a single buffer which may overflow along a packet's path. Moreover, the bridge node shares a ring with the source. So we are able to provide feedback to all the sources on the ring, in the form of a circulating *quota-matrix*, which allows them to slow down when bridge buffers are close to capacity.

5.1 Buffers and Quotas

We start the discussion of the flow control protocol by describing the various buffers each node holds, and the quotas that govern each buffer's service rate.

We assume that a packet is identified as either *LOCAL* or *REMOTE* by the routing algorithm, which also determines which ring R_i the packet should be placed on. Once this routing decision is made, the packet is placed in the appropriate buffer. Each node u has two types of buffers (queues) for host-generated traffic that is designated to a ring R_i:

1. LQ_i for host-to-ring *LOCAL* traffic. u has one such buffer for each ring it belongs to.

2. $RQ_{ij}(x)$ for host-to-ring *REMOTE* traffic that is designated to ring R_j via the bridge node x. Thus u has $(n-1)(d-1)$ RQ_{ij} buffers for ring R_i, one for each remote ring

R_j that can be bridged to via a neighbor x (and a total of $d(n-1)(d-1)$ such buffers for all the rings that u belongs to).[3]

Note that both LQ and RQ are unbounded queues.

In addition, a node u also needs to bridge ring-to-ring traffic. For this u has a buffer BQ_{ij} in which it stores $REMOTE$ packets it bridges from R_i and R_j. Thus u has $d-1$ such BQ buffers for every ring R_i it belongs to, and $d(d-1)$ BQ buffers in total. The BQ buffers are bounded, and their size (in packets) is denoted by $MAXBQ$. We shall prove that the flow control is congestion-free regardless of the value of $MAXBQ$.

The flow control is done using a system of quotas that govern the service rates for the different queues. These quotas are in units of packets that a node u can transmit during a cycle rotation time D. Since $LOCAL$ traffic does not pose a buffer-overflow problem we do not need to impose a specific quota on the LQ buffer (other than the general access-control quota k). Thus we have the following quotas: A node u is allowed to transmit at most $r_u^{ij}(x)$ packets per cycle out of buffer $RQ_{ij}(x)$.

The r^{ij} quotas are modified adaptively by the flow control algorithm. Intuitively, as a buffer BQ_{ij} at some bridge node x is filled, the bridge adaptively reduces the quotas $r_u^{ij}(x)$ of all the nodes u on ring R_i. The next sections describe the details of the mechanisms involved.

Remark: The quotas of the buffer service rates are not meant to guarantee fairness among the various traffic types that a node handles, which is why we do not propose using techniques such as the window-based access scheme of Section 4 for these quotas. Furthermore, we do not specify the policy that determines which packet should be selected from those ready in the LQ, RQ, and BQ buffers, or how these buffers are managed. Many policies that guarantee fair queueing have been suggested in the literature (e.g., [Nag87, DKS89, SV96]). A recent treatment of the buffer management issues can be found in [AMO97].

5.2 The Quota Matrix Mechanism

Consider now a node u in its role as a bridge node between R_i and R_j. To avoid dropping packets we need to control the total arrival rate to the bridge buffer BQ in the current cycle as a function of the BQ size at the end of the previous cycle. Informally, the total arrival rate at u's BQ_{ij} buffer is closely related to the sum of remote quotas on the ring R_i, $\sum_{x \in R_i} r_x^{ij}(u)$, thus decreasing the r^{ij} quotas should decrease the arrival rate at BQ_{ij}. However the $r_x^{ij}(u)$ quotas are local variables at each node x. Thus a mechanism is needed to inform the nodes

[3] In the GQ constructions there is always a single node x bridging between rings R_i and R_j. However in general there may be several possible bridges, which we emphasize by explicitly noting the bridge node x in the queue name.

when quota adjustments are needed on some ring, and we need to account for the delay in propagating the updated quotas to the nodes on the ring.

The mechanism we propose is based on circulating a *quota matrix* Q^i on each ring R_i. Each node u advertises the total number of packets that it is willing to bridge from R_i to R_j in the entry $Q^i[u, R_j]$. If there is a unique bridge node on R_i for every R_j that can be bridged to, then Q^i has $n(d-1)$ entries in total. We need to ensure that the following *flow control* condition holds when the quota matrix *returns* to node u:

$$\sum_{x \in R_i} r_x^{ij}(u) \leq Q^i[u, R_j]. \tag{1}$$

When a node x receives the quota matrix Q^i on ring R_i, x adjusts its *REMOTE* quotas by setting

$$r_x^{ij}(u) \leftarrow \left\lfloor \frac{Q^i[u, R_j]}{n-1} \right\rfloor \tag{2}$$

for all rings R_j that can be bridged to from R_i. Then x needs to calculate the new advertisements for the number of packets *it* is willing to bridge from ring R_i in the next cycle, update the entries it is in charge of in the Q^i matrix (namely the entries $Q^i[x, R_j]$ for all R_j that x belongs to), and forward the Q^i matrix. It is obvious that condition (1) is met when all the nodes use (2) to update their r^{ij} quotas.[4]

For the purpose of flow control, the circulation of the Q matrix gives a natural definition of cycles, as follows.

Definition 5.1 *A* cycle *for node u on ring R_i starts when u transmits the Q^i matrix and ends when it receives it back.*

Note that the cycles of Definition 5.1 start at different times for different nodes. Thus when a node u advertises that it is willing to bridge B packets from R_i to R_j during the next cycle, this is with respect to cycles as perceived by u. However the quota $r_x^{ij}(u)$ at node x on the same ring R_i is applied to cycles as perceived by x.

5.3 Updating the Quotas

Consider a particular buffer BQ_{ij} at a node u. Let B_m denote the number of packets that u advertises at the *start* of cycle m (i.e., the value that u places in $Q^i[u, R_j]$), and let $|BQ_m|$ be the number of packets in the BQ_{ij} buffer at the *end* of cycle m (see Figure 2). Recall

[4]Rule (2) is an over-simplified method for distributing the total advertised capacity to the individual nodes, and may lead to under-utilization of the ring bandwidth. E.g., when $Q^i[u, R_j] < n$ then all the nodes will set $r_x^{ij}(u) \leftarrow 0$, when in fact some of them should be able to send packets. We omit the details of more careful distribution methods for sake of brevity.

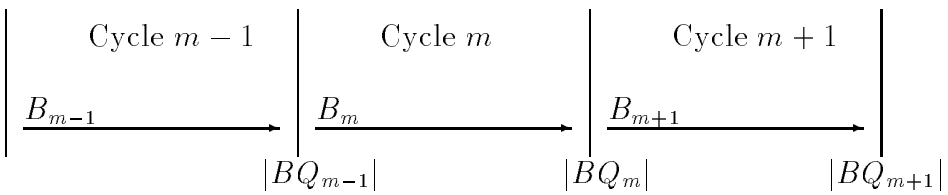

Figure 2: Cycles, bridge queue sizes and advertised capacities, from the bridge node's point of view.

that the maximal size of $|BQ_{ij}|$ is $MAXBQ$. The following rule is used to update the new advertisement.

$$B_{m+1} \leftarrow MAXBQ - |BQ_m| - B_m, \tag{3}$$

which is simply the difference between the current free space and number of packets committed to at the start of the previous cycle. The combined access control, flow control and routing protocol is sketched in Figure 3.

5.4 The Flow Control Protocol is Congestion-free

We start the analysis of the flow control protocol by a simple example that illustrates the first 3 cycles following the system start up as seen by a bridge node u on ring R_i. This example will also serve as the base case for the proofs to follow.

Example 5.2 Assume that $MAXBQ = 10$.

0. The system starts up with $r^{ij} = 0$ and all the buffers are empty, so $|BQ_0| = 0$. For notational convenience assume also that $B_0 = 0$.

1. At the beginning of cycle 1, u advertises $B_1 = MAXBQ - |BQ_0| - B_0 = 10$. Note that at the end of cycle 1 we still have $|BQ_1| = 0$ since no packet can overtake the Q matrix, and the initial r^{ij} quotas were all 0.

2. At the beginning of cycle 2 the buffer is still empty, yet u advertises $B_2 = MAXBQ - |BQ_1| - B_1 = 0$ since packets may be in flight. Packets that are received during u's cycle 2 were all transmitted by other nodes during u's cycle 1—so there can be at most $B_1 = 10$ of them. Suppose that only 5 were sent, and that 2 were already forwarded onto ring R_j, leaving $|BQ_2| = 5 - 2 = 3$ at the end of cycle 2.

3. Thus at the start of cycle 3, u advertises $B_3 = MAXBQ - |BQ_2| - B_2 = 10 - 3 - 0 = 7$, etc.

Lemma 5.3 $|BQ_{m+1}| - |BQ_m| \leq B_m$.

Proof: Packets sent under quota B_m will appear at u only *after* the quota matrix returns to u at the end of cycle m, thus they will be counted in u's cycle $m+1$. Hence the total

- Upon receiving a *FULL* slot containing $P(v)$:
  1. If $v = u$, remove $P(v)$ from R_i, mark the slot *EMPTY*, go to procedure for *EMPTY* slots.
  2. Else if u is a bridge for $P(v)$, remove $P(v)$ from R_i, mark the slot *EMPTY*, store $P(v)$ in the appropriate $BQ_{ij}(u)$ buffer.
  3. Else forward $P(v)$ on R_i.
- Upon receiving an *EMPTY* slot:
  1. Verify that the access control quota k has not been exceeded.
  2. Select a packet $P(v)$ from LQ_i, some BQ_{ji}, or some $RQ_{ij}(x)$ for which the flow control quota $r_u^{ij}(x)$ has not been exceeded.
  3. Mark the slot *FULL*, send $P(v)$ on R_i.
- Upon receiving the Q^i matrix:
  1. Adjust the $r_u^{ij}(x)$ quotas.
  2. Update the $Q^i[u, R_j]$ entries.
  3. Forward Q^i on R_i.

Figure 3: The combined protocol for node u on ring R_i.

number of arrivals at BQ_{ij} during cycle $m+1$ is at most B_m. Therefore even if no packet is forwarded onto R_j (i.e., no packet leaves BQ_{ij}) during cycle $m+1$, the change in the buffer's occupancy during this cycle is at most the capacity B_m advertised at the start of the previous cycle. ∎

Theorem 5.4 *$|BQ_{m+1}| \leq MAXBQ$ for all cycle m.*

Proof: For $m = 0, 1$ the theorem holds by Example 5.2. For $m \geq 2$ we apply Lemma 5.3 (twice) and rule (3) to obtain that

$$\begin{aligned} |BQ_{m+1}| &\leq |BQ_m| + B_m \\ &= |BQ_m| + (MAXBQ - |BQ_{m-1}| - B_{m-1}) \\ &= MAXBQ + (|BQ_m| - |BQ_{m-1}| - B_{m-1}) \\ &\leq MAXBQ + (B_{m-1} - B_{m-1}) \quad = \quad MAXBQ. \end{aligned}$$ ∎

We proved that there is no buffer overflow at the end of each cycle. However it is easy to see that at intermediate time points within the cycle the buffer cannot overflow either, since the advertised capacity computed by rule (3) is conservative and does *not* assume that any packets leave the buffer. In fact even if no packets ever leave the buffer, it will not overflow—the flow control will quench all the sources sending via this buffer by setting their r^{ij} quotas to 0.

Remark: For simplicity in the proof we assumed (in Example 5.2) that $B_0 = 0$. However our simulation studies show that using this initial setting leads to a long period of oscillations in the quotas, in which nodes alternate between advertising a capacity $B_m = MAXBQ$ and a capacity $B_m = 0$. Instead, our simulations show that the oscillation is eliminated and the quotas stabilize much faster if we set $B_0 \leftarrow MAXBQ/2$. It is easy to see that this is safe when the network performs a "cold start" and no packets are in flight.

6 Properties of the Networks

The combination of the one-bridge property, the combinatorial properties of GQ designs, and our flow control protocols gives a network which has many desirable characteristics. In this section we show that our networks enjoy a bounded network delay which is independent of the traffic pattern; a high degree of path redundancy which translates to excellent network survivability; a self-routing property which simplifies the path selection algorithm and allows easy load-balancing; and lower link costs in comparison to earlier proposals.

6.1 Bounded Network Delay

Let $\Delta(uv)$ denote the network delay from node u to v. Recall that D is the time it takes for a slot to rotate all the way around a single ring in the network. The network delay for *LOCAL* packets is clearly bounded by the propagation delay on a ring:

Fact 6.1 *If u and v are neighbors on some ring R_i then $\Delta(uv) \leq D$.*

When a bridge node x sees an *EMPTY* slot on ring R_j, it needs to select which packet to put in it from the packets waiting in its LQ_j, RQ_{ji}, and BQ_{ij} buffers. The bound on the network delay for *REMOTE* packets depends on the guarantees that can be given for the packet selection policy used at node x (and on the rotation time D).

Proposition 6.2 *Assume that u sends REMOTE packets to v via ring R_i and that they are bridged to R_j by node x. If the packet selection policy at x guarantees that at least b packets will be selected from BQ_{ij} in each cycle then*

$$\Delta(uv) \leq D(2 + \lceil MAXBQ/b \rceil).$$

Proof: From the one-bridge property we know that the network delay for a remote pack consists of the delay on R_i, the queueing delay in BQ_{ij} at x, and the delay on R_j. Therefore

$$\Delta(uv) \leq \Delta(ux) + D\lceil |BQ_{ij}|/b \rceil + \Delta(xv) \leq D(2 + \lceil MAXBQ/b \rceil). \qquad \blacksquare$$

Remark: It is easy to give such a guarantee b for many fair scheduling policies. For instance, a simple round-robin policy which rotates equally among the single LQ_j buffer, the $(n-1)(d-1)$ RQ_{ji} buffers and the $d-1$ BQ_{ij} buffers guarantees that $k/(nd-n+1)$ packets would be selected from each BQ_{ij} buffer each cycle (recall the k is the access control quota for all types of packets).

6.2 Path Redundancy, Survivability and Cost

A GQ-based network of degree d has a high degree of path redundancy: Proposition 2.4 shows that there are d edge-disjoint one-bridge paths between every two non-neighboring nodes u and v. In fact u can choose to send a packet $P(v)$ to v on any ring $R_i \ni u$ it belongs to, and a bridge $x \in R_i$ is guaranteed to exists to relay the packet onto a ring R_j that v belongs to.

This combinatorial property gives us a "self-routing" capability: In principle the source does not need to know the identity of the bridge node x or the destination ring R_j. Note, though, that our combined routing and flow control protocol does *not* use this self-routing capability. In our protocol u needs to place $P(v)$ in a particular $RQ_{ij}(x)$ buffer, that is

subject to the respective $r_u^{ij}(x)$ quota, and for this u needs to know the identities of the bridge x and the ring R_j.

However, our protocols can use the path redundancy to balance the traffic load on the rings. This is done by a *path selection policy* which we do not specify in this paper. In the simulation studies (Section 8) we implemented a simple round-robin path selection policy.

The same path redundancy also implies desirable survivability properties for the network. It is easy to see that the network can survive $d-1$ link or node failures without becoming disconnected. For *REMOTE* packets, essentially the same routing protocol can be used even when some links or nodes fail.

LOCAL packets would require slightly different routing in failure mode. A *LOCAL* packet $P(v)$ may need to be sent to some other node z which shares a (still functioning) ring R_j with the destination v, and z would then forward the packet to v. Note that this bypass route traverses three rings and two bridges (one en-route to z and z itself). For the first part of the route the packet $P(v)$ would need to be converted to a special *REMOTE* packet.

Using a simplistic model of cost we can see that GQ-based networks achieve their desirable properties quite cheaply. If we associate a unit of cost for each link, then a GQ-based network with $M = N$ rings of size $O(N^{1/3})$ would have a cost of $O(N^{4/3})$. This is only slightly more than the minimal cost of $N-1$ that is required for basic network connectivity, and is less than the $O(N^{3/2})$ cost of the BIBD-based networks of [YOY94].

7 Construction of Networks of Arbitrary Size

In Section 2 we presented the basic methods for combinatorial construction of GQ-based networks. In particular, to construct a network of N nodes with degree d we need a GQ of order (s,t) that has N nodes such that $d = t+1$. However recall that a GQ always has the constraint that $N = (s+1)(st+1)$. Therefore not every integer N is suitable as a GQ size. Moreover, GQ designs are not even known for all N's that *do* satisfy the constraints. Thus, the number of GQ designs that we can apply directly is quite limited (see Table 2 for a list of designs that are available).

Given a positive integer N of network nodes, and a degree constraint d, it may not be feasible to construct a network that satisfies both constraints. In this section, therefore, we generalize the basic design methods and show how to construct networks of arbitrary size while keeping the degree of each node fixed.

Our approach is based on finding a GQ(s,t), called the *base-design*, which has fewer than N nodes and in which the degree bound d is not exceeded (i.e., $t+1 \leq d$). We then scale the base design up to N nodes without increasing the node degrees.

Note that a GQ design can be described by an *incidence matrix* A which has the rings as its columns and the nodes as its rows. An entry $a_{i,j}$ of the incidence matrix A is equal to one if the i'th node resides in the j'th ring, otherwise it is equal to zero.

7.1 The Scaling Operations

The following operations were introduced in [YOY97], where they were used to scale projective planes to arbitrary sizes. Here we use them to scale GQ designs to any desired network size.

7.1.1 Insertion

The insertion operation is defined between a base design (GQ) and a positive integer L. The insertion adds L nodes to the design by modifying its incidence matrix A. An insertion operation consists of (1) adding of a new row to A, (2) selecting an existing row i, and (3) copying the $a_{i,j}$ values of the row i into the new row. Thus the node that corresponds to row i of A is *cloned* to form the new node being inserted. To insert L new nodes this procedure is repeated L times.

Clearly inserting nodes does not change the maximum node degree or the total number of rings. However the rings do increase in size since a cloned node resides on all the rings its originator resided on.

Note that there is a large degree of freedom in the choice of the row i to be copied in step (2) (i.e., in the choice of the node to be cloned). As in [YOY97], we suggest using this freedom to minimize the maximal resultant ring size.

7.1.2 Multiplication

This multiplication operation is defined between a GQ design of N nodes and a positive integer L. The result of the multiplication is a scaled design of LN nodes.

In terms of the incidence matrix A, the multiplication operation consists of repeating *all* the rows of A L times. In other words, each node in the design is cloned into L duplicate nodes which are inserted into the design.

7.2 The Scaling Algorithms

The scaling operations identified above can be used by an algorithm to clone some nodes of a design to reach the desired network size. In general, though, deciding which are the

optimal nodes to clone so that the maximum ring size is minimized is a very difficult problem [Col98]. Instead, we suggest two fast (sub-optimal) scaling algorithms.

The first possibility is to use the greedy approach of [YOY97]. In this approach nodes are inserted one by one until the desired network size is reached. In each insertion step the algorithm clones a node which will result in the minimal increase in the maximal ring size.

An alternative is to use a randomized approach. In this approach the nodes to be cloned are selected uniformly at random. Many such randomly scaled designs are generated, and the best one among them is selected as the output.

8 The Simulation Study

In order to evaluate the performance of our protocols on networks that enjoy the one-bridge property, we conducted a simulation-based study. For this purpose a discrete event simulator program was written (in C++). We ran the simulator using a variety of networks and under many traffic load conditions.

8.1 Description of the Simulation Model

The basic unit of time in the simulation is a clock tick. In the simulator's basic mode of operation, at each clock tick each node in the network generates new traffic, receives the content of a slot, processes it, and possibly selects and sends a packet using the same slot. At the end of the clock tick the slots on all the rings rotate one position clockwise.

In the networks we implemented the nodes are equivalent, the rings are equivalent, and the nodes are symmetrically arranged on each ring. In particular the number of inter-node clock ticks between any two consecutive nodes on a ring is the same, and is a simulator parameter denoted by it.

The basic mode of operation represents the network's *base bandwidth*. We are also able to simulate higher bandwidths, as long as they are integer multiples of the base bandwidth. This is implemented by using an integer *bandwidth factor* (bw). So in fact during each clock tick each node processes bw slots, and $bw = 1$ indicates the base bandwidth with one slot per clock tick. The total number of slots on a ring of size n is $p = n \times it \times bw$. The access control quota that each node has is therefore $k = p/n = it \times bw$.

A node's packet selection policy, for each ring R_j it belongs to, is a round-robin policy. The policy rotates equally among the single LQ_j buffer, the $(n-1)(d-1)$ RQ_{ji} buffers and the $d-1$ BQ_{ij} buffers.

An important parameter to the simulator is the maximal size of the BQ buffers ($MAXBQ$).

The LQ and RQ buffers were implemented as unbounded.

8.1.1 The Traffic Model

To simulate bursty data traffic we use an "on/off" traffic model, which is controlled by two parameters: the burst probability P_b and the maximum burst length M_b. Every clock tick each node u flips an independent probability-P_b coin for each other node v. If the coin shows "heads" u picks a value $1 \leq \ell \leq M_b$ uniformly at random, and generates a burst of ℓ packets with destination v. Thus the average number of packets that are injected into the network each clock tick is $E[Traffic] = N(N-1) \times P_b \times M_b/2$.

Path selection is done burst-by-burst, so all the packets that belong to a single burst follow the same path through the network. If the source u and destination v are neighbors on some ring R_i ($LOCAL$ bursts) then the whole burst is placed in u's LQ_i buffer. As for $REMOTE$ bursts, recall that there are d disjoint one-bridge paths between u and v in a GQ-based network, each corresponding to a different RQ_{ij} buffer. To balance the load on the different rings, we again use a round-robin policy, this time for path selection: u cycles among the d possible paths it has to v, burst by burst.

	Load/Bandwidth	**Buffer Size**	**Scaling**
Traffic parameters			
Maximum burst size (M_b)	20		
Burst probability (P_b)	* .0001–.02	* .0005, .0006, .00765	* .0037, .0012, .0004, .00075
Network parameters			
Number of nodes (N)	45	45	* 15, 40, 156, 400
Ring size (n)	3	3	* 3, 4, 6, 8
Degree (d)	5	5	* 3, 4, 6, 8
Bandwidth factor (bw)	* 1, 2, 3	2	1
Inter-node clock ticks (it)	20		
Slots per ring (p)	* 60, 120, 180	120	* 60, 80, 120, 160
Buffer Management			
Packet selection policy	round-robin		
$MAXBQ$	20	* 3–100	* 20, 30, 50, 70

Table 3: Simulation model parameters for the reported studies. Value ranges prefixed by a "$*$" indicate parameters that were varied in each study.

8.1.2 The Performance Measures

We gathered statistics on the following types of performance measures. For each of them we calculated the mean, range, and standard deviation. The curves in the sequel all show the mean values.

The *ring utilization* is the fraction of slots containing a packet out of the total number of slots.

The *waiting time* of a packet in a buffer is the number of clock ticks from the moment the packet is placed in the buffer until it is placed in a slot on the ring.

The *network delay* of a packet is the number of clock ticks from the moment a packet is placed in a slot at its source until it reaches its destination. For *REMOTE* packets this includes the waiting time in the bridge's *BQ* buffer.

Each point in our graphs represents the output of a run of 12,000 clock ticks. Transients [Jai91] were eliminated by discarding the first 2000 clock ticks. For such runs, the graphical height of the 95%-confidence intervals[5] was very close to the size of the symbols depicted on the curves. Therefore in order to reduce the complexity of our graphs, we do not show these confidence intervals.

8.2 Results and Interpretations

8.2.1 The Load/Bandwidth Study

In this study we show the network performance under increasing traffic loads, for three bandwidths. We performed the same study using several different networks, all of which gave qualitatively similar results. Here we report the results obtained on a network based on the so called $Q(5,2)$ GQ [CD96]. The parameters we used are listed in Table 3.

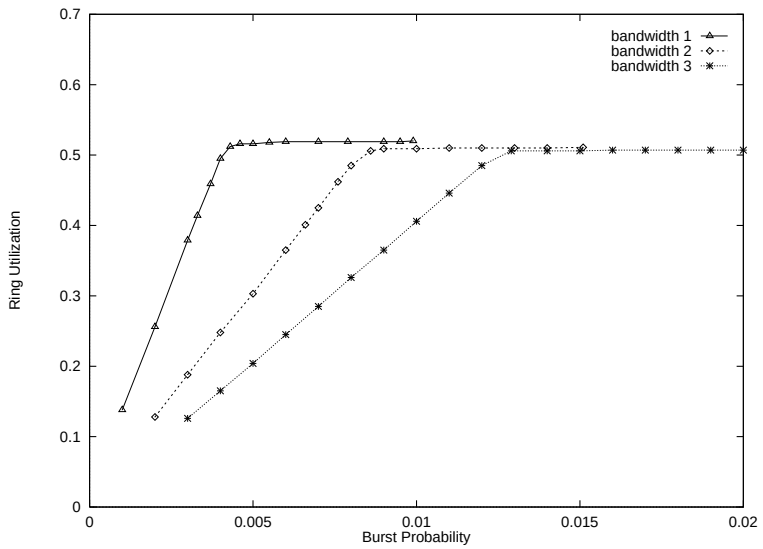

Figure 4: Ring utilization, Load/Bandwidth study

[5] A 95% confidence interval means that that value appears within the specified interval with probability 0.95. See [Jai91] for a precise definition of confidence interval.

We see in Figure 4 that as the traffic load increases the ring utilization grows linearly up to a "knee point", where the utilization levels off at ≈50%. This fits the prediction of Proposition 4.9 very well: The packet destinations are chosen uniformly in our traffic model, so $E[PLen] = p/2$. This behavior indicates that the bottleneck in emulating a ring protocol is the access control rather than the flow control protocol, since Proposition 4.9 is a consequence of the access control.

Note, though, that the utilization levels off slightly above 50% (e.g., $Util = 0.519$ for $bw = 1$). This is since each ring has a quota matrix which always occupies one slot, which Proposition 4.9 does not account for.

When the bandwidth increases, the network reaches its maximal utilization at higher traffic loads: the knee point is 0.0043, 0.0086, 0.0129 for bandwidth factor $bw = 1, 2, 3$.

We remark that the well understood tradeoff between fairness and efficiency and optimality holds here. If we relax the fairness constraint and over-allocate the bandwidth (i.e., do statistical multiplexing) or allow techniques like slot-reuse we can easily drive the utilization much higher.

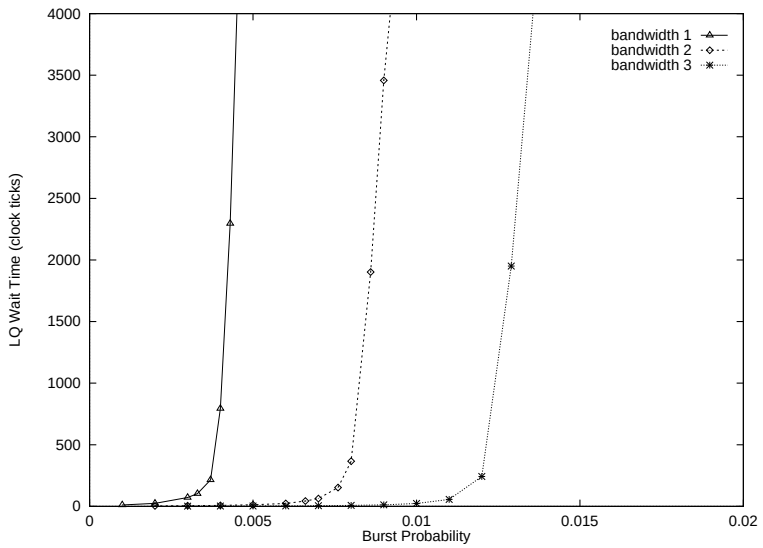

Figure 5: Waiting time for $LOCAL$ packets in the LQ buffers, Load/Bandwidth study.

The waiting time for $LOCAL$ packets in the LQ buffers is shown in Figure 5. Since the protocols are congestion-free, when the network becomes saturated the access control causes the LQ buffers to explode and the waiting time increases rapidly. The data shows that beyond the saturation point the network is in an unstable state and the LQ buffers are growing as a function of time.

Figure 6 compares the behavior of the RQ and LQ buffers. We see that when the network is not saturated ($P_b \leq 0.0129$) packets wait a shorter time in an LQ buffer than in an RQ buffer (e.g, 23.2 ticks versus 145.3 ticks when $P_b = 0.01$). This is since $REMOTE$ packets

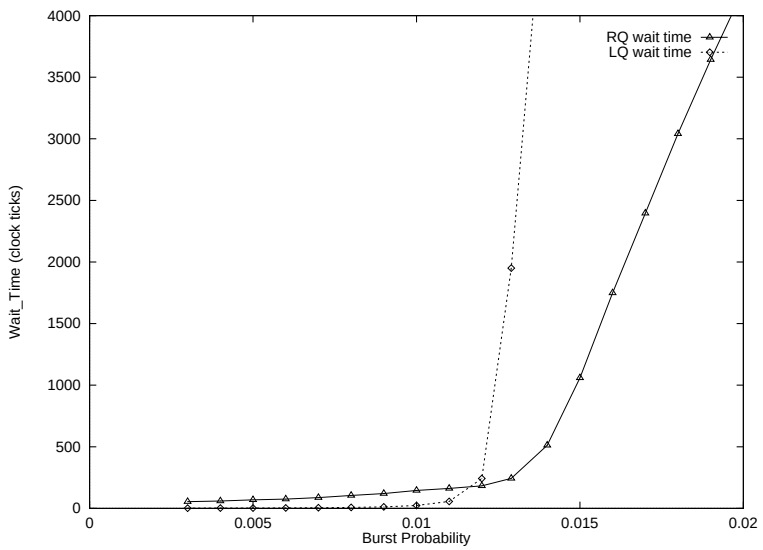

Figure 6: The waiting time of *LOCAL* versus *REMOTE* packets, for bandwidth factor 3, Load/Bandwidth study.

are delayed by the flow control protocol whereas *LOCAL* packets are only delayed by the access control.

Despite this, the *RQ* buffers can sustain a higher traffic load before exploding. This is somewhat surprising at first glance since most of a node's destinations are *REMOTE* (in the $Q(5,2)$ network a node has 12 neighbors and 32 non-neighbors). But this is mitigated by the fact that a node has *many RQ* buffers. The LQ_i buffer sees traffic destined to the $n-1$ nodes on R_i. In comparison, packets placed in RQ_{ij} are destined to one of the $n-1$ nodes on R_j—but there are d disjoint paths to each of them, each with a *different RQ* buffer at the source. Since we use round-robin path selection, any specific RQ_{ij} buffer thus sees only $1/d$ of the traffic that an *LQ* buffer sees. The knee point for the *RQ* buffers is much lower than at $d = 5$ times the maximum traffic load for the *LQ* buffer due to the extra delays caused by the flow control.

In Figure 7 we see the average network delay for *REMOTE* packets. This delay consists of the propagation delay on each of the two rings and the waiting time in the bridge's BQ_{ij} buffer. The propagation delays contribute $2 \times E[PLen]$ clock ticks (60 ticks in this case), and any additional delay is due to the wait in a *BQ* buffer. Note that the average delay is much less than the bound of Proposition 6.2: In the $Q(5,2)$ network the round-robin packet selection guarantees that $b = 20/11$ packets will be removed from each *BQ* buffer in each cycle, so Proposition 6.2 gives a bound of 780 ticks, whereas the maximal average delay we observe is 238 ticks (for $bw = 1$).

LOCAL packets are delayed only $E[PLen] = 30$ ticks regardless the of traffic load since they do not pass a bridge and so they do not suffer from any waiting time (graph omitted).

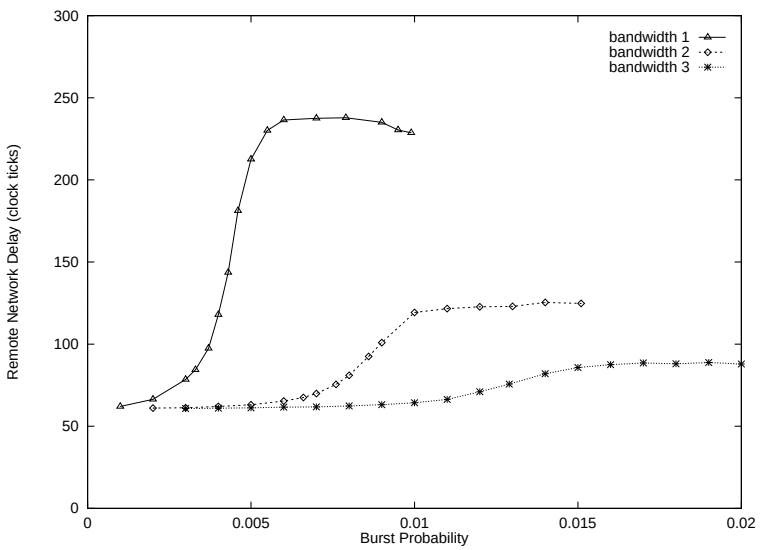

Figure 7: Network delay for *REMOTE* packets, Load/Bandwidth study.

8.2.2 The Buffer Size Study

In this study we tested the effect of the size of the *BQ* buffers on the network performance, again using the $Q(5,2)$ network. Table 3 gives the parameter settings we used.

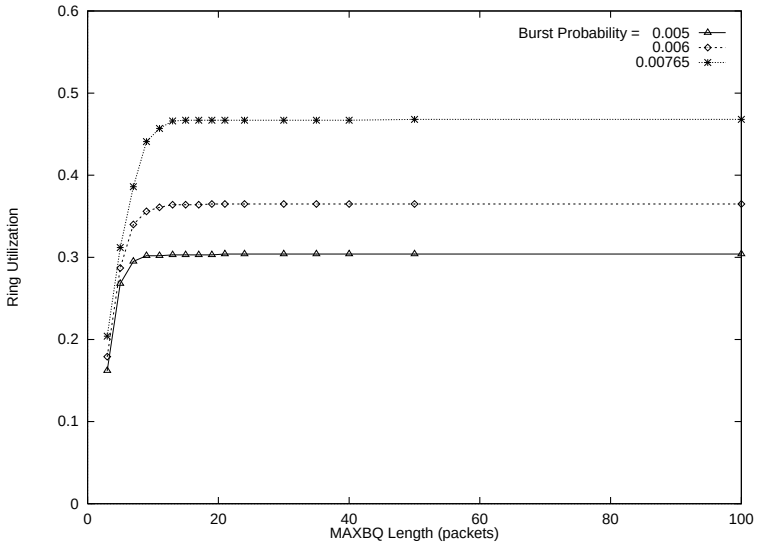

Figure 8: Ring utilization, Buffer Size study

Figure 8 shows that the ring utilization is insensitive to the actual *MAXBQ* value—as long as it is not too small. When *MAXBQ* is too small the flow control protocol denies access to *REMOTE* packets, thus nodes are unable to use all of their access control quota. This causes the rings to be under-utilized, and the *REMOTE* delay to be very high. When *MAXBQ* is extremely low (below 7 in this case) the data shows that the *RQ* buffers become

unstable. However having $MAXBQ \geq 15$ does not increase the utilization. A reasonable rule-of-thumb seems to be $MAXBQ \geq (n-1) \times M_b/2$, i.e., allowing each neighboring node on the source ring to generate one average-sized burst per cycle.

8.2.3 The Scaling Study

In this study we examined the scalability of our networks. For this we ran the equivalent of the Load/Bandwidth study on 4 networks that belong to the $W(q)$ [CD96] family of GQs and have $n = d = O(N^{1/3})$. The parameter settings are listed in Table 3.

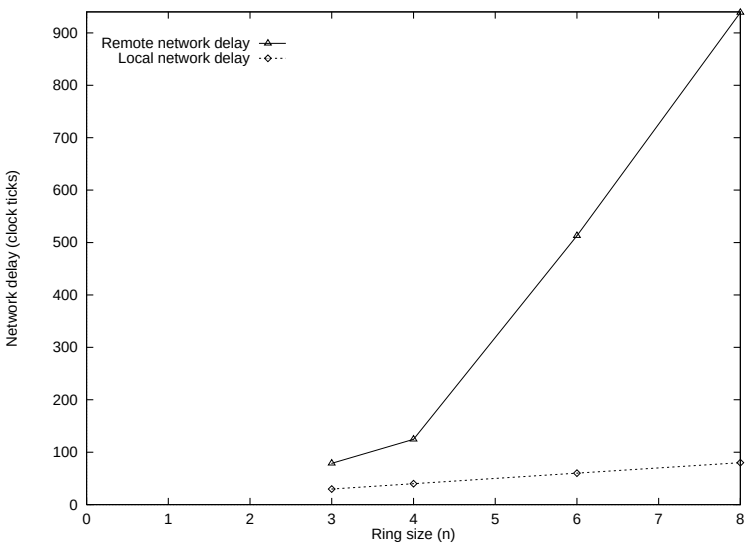

Figure 9: Network delay as a function of ring size for *LOCAL* and *REMOTE* packets.

To make the performance of different sized networks comparable we needed to pick "equal" traffic loads for the various networks. However note that in our traffic model, for a fixed burst probability P_b, the total number of packets injected per cycle increases at a rate of $\Omega(N^2)$ as a function of the network size N. Therefore we chose to compare the performance of the networks for *different* values of P_b. We selected P_b values which yielded ring utilizations of 40–41% on their respective networks. A 40% utilization represents the performance of a near-saturated network, since a 50% utilization is the theoretical maximum.

Figure 9 shows the network delay as a function of the ring size. As expected, the delay experienced by *LOCAL* packets is a linear function of the ring size. However *REMOTE* packets experienced a super-linear increase in the delay, which is due to longer waiting times in the BQ buffer. This is not caused by the $MAXBQ$ value—picking smaller values only hurt the utilization (as we have seen above) and larger values did not make a noticeable difference. Rather, the cause is the round-robin packet selection policy: Each BQ buffer is guaranteed a $1/(nd - n + 1)$ fraction of the node's quota, and when $n = d$ this decreases as $O(1/n^2)$. So as the network size increases, packets remain longer in the BQ buffers.

A promising packet-selection policy that would avoid these additional network delays is the following: Call packets in LQ and RQ "new" and call packets in BQ "in-flight". Then a 2-level round-robin policy, the higher level alternating between new and in-flight packets, and the lower level performing round-robin within each category, would guarantee a $1/2(d-1) = O(1/n)$ fraction of the quota to each BQ buffer, so one can hope that the increase in the BQ waiting time (and thus the network delay) would become linear in the ring size n.

9 Conclusions

We have described a new methodology for the design of multi-ring networks, which is based on combinatorial design theory. We capitalized on the designs' combinatorial properties to construct control protocols with provable guarantees: fair, congestion-free, bounded-delay routing; small ring size and low node degree; fixed internal buffer sizes; high survivability; and low cost. We also evaluated the networks' performance by extensive simulations and found them to scale up well and to behave much better than their worst-case guarantees in the face of bursty traffic. We believe that our designs offer significant benefits for future network infrastructures.

Acknowledgments

The simulator program was skillfully written by Yueyue Song of Rutgers University.

References

[AMO97] Y. Afek, Y. Mansour, and Z. Ostfeld. Space efficient fair queuing by stochastic memory multiplexing. Technical Report 322/97, Dept. of Computer Science, Tel-Aviv University, 1997.

[ANS88] ANSI T1.105-1988. American National Standard for Telecommunications, Digital Hierarchy—Optical Interface Rates and Format Specifications, 1988.

[BCK+83] W. Bux, F. H. Closs, K. Kummerle, H. J. Keller, and H. R. Mueller. Architecture and design of a reliable token-ring network. *IEEE Journal of Selected Areas in Communications*, SAC-1(5):756–765, November 1983.

[BOY97] M. Baldi, Y. Ofek, and B. Yener. Adaptive real time group multicast. In *IEEE INFOCOM'97*, 1997.

[CD96] C. J. Colbourn and J. H. Dinitz. *The CRC Handbook of Combinatorial Designs*. CRC Press, Boca Raton, 1996.

[Col98] C. J. Colbourn. Personal communication, 1998.

[Dan98] P. H. Dana. Global positioning system overview. The Geographer's Craft Project, Department of Geography, The University of Texas at Austin, 1998. `http://www.utexas.edu/depts/grg/gcraft/notes/gps/gps.html`.

[DH98] B. Doshi and P. Harshavardhana. Broadband network infrastructure of the future: Roles of network design tools in technology deployment strategies. *IEEE Communications Magazine*, 36(5):60–71, 1998.

[DKS89] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proc. ACM SIGCOMM*, pages 1–12, 1989.

[DP91] A. De and N. Prithviraj. Ring-connected-ring (RCR) topology for high-speed networking: Analysis and implementation. *ACM SIGCOMM Computer Communications Review*, 21(3):33–44, July 1991.

[FN69] W. D. Farmer and E. E. Newhall. An experimental distributed switching system to handle bursty computer traffic. *Proc. ACM Symp. on Problems in Optimization of Data Communication Systems*, pages 1–33, 1969.

[Jai91] R. Jain. *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, 1991.

[Kap96] E. D. Kaplan. *Understanding GPS, Principles and Applications.* Artech House, Boston, 1996.

[LOY96] C-S. Li, Y. Ofek, and M. Yung. "Time-driven priority" flow control for real-time heterogeneous internetworking. In *IEEE INFOCOM'96*, 1996.

[LSBW98] E. Livermore, R. P. Skillen, M. Beshai, and M. Wernik. Architecture and control of an adaptive high-capacity flat network. *IEEE Communications Magazine*, 36(5):106–112, 1998.

[MB76] R. M. Metcalfe and D. R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, July 1976.

[Nag87] J. Nagle. On packet switches with infinite storage. *ACM/IEEE Transactions on Networking*, 35(4):435–438, 1987.

[PT84] S. E. Payne and J. A. Thas. *Finite Generalized Quadrangles.* Research Notes in Mathematics 110. Pitman, London, 1984.

[RGA85] C. S. Raghavendra, M. Gerla, and A. Avižienis. Reliable loop topologies for large local computer networks. *IEEE Trans. Computing*, C-34, No. 1:46–55, January 1985.

[RS86] C. S. Raghavendra and J. A. Silvester. A survey of multi-connected loop topologies for local computer networks. *Computer Networks and ISDN Systems*, 11:29–42, 1986.

[SV96] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round robin. *ACM/IEEE Transactions on Networking*, 4(3):375–385, 1996.

[Tan96] A. S. Tanenbaum. *Computer Networks.* Prentice Hall PTR, 3rd edition, 1996.

[YOY94] B. Yener, Y. Ofek, and M. Yung. Topological design of loss-free switch-based LANs. In *IEEE INFOCOM'94*, 1994.

[YOY97] B. Yener, Y. Ofek, and M. Yung. Combinatorial design of congestion-free networks. *ACM/IEEE Transactions on Networking*, 5(6):989–1000, December 1997.