DEADLOCK-FREE PACKET SWITCHING NETWORKS WITH VARIABLE PACKET SIZE

H.L.Bodlaender

RUU-CS-85-25
September 1985

Rijksuniversiteit Utrecht

Vakgroep Informatica

Budapestlaan 6 3584 CD Utrecht
Corr. adres: Postbus 80.012 3508 TA Utrecht
Telefoon 030-53 1454
The Netherlands

DEADLOCK-FREE PACKET SWITCHING NETWORKS WITH VARIABLE PACKET SIZE

H.L.Bodlaender

Technical Report RUU-CS-85-25
September 1985

Department of Computer Science
University of Utrecht
P.O. Box 80.012, 3508 TA Utrecht
the Netherlands

DEADLOCK-FREE PACKET SWITCHING NETWORKS WITH VARIABLE PACKET SIZE

H.L. Bodlaender*

Department of Computer Science, University of Utrecht
P.O. Box 80.012, 3508 TA Utrecht, the Netherlands.

Abstract. In packet switching networks the occurence of deadlock is very undesirable. Therefore one needs controllers (algorithms that control the flow of packets through the network), that prevent the occurrence of deadlock. We present a class of controllers that prevent deadlock, use only local information known to the processors and allow packets to have a variable size. Some controllers in this class are proven to be optimal with respect to other controllers using the same local information.

1. Introduction.

Consider a packet switching network, represented by a (possibly directed) graph G=(V,E). V represents the set of processors, E represents the set of links between processors. Some processors may want to send messages (packets) to other processors in the network. Each packet has to follow a path in G from its source node to its destination node, in accordance to some routing strategy. We make but one assumption on the routing strategy that is followed by a network: there must be an integer k, known to all processors, such that each packet needs to traverse at most k links to arrive at its destination. Note that k is not necessarily at least the diameter of G: it is possible that we do not want to send messages between nodes that have a large distance between them.

During the transportation of a packet from source to destination,

*This work was supported by the Foundation for Computer Science (SION) of the Netherlands Organization for the Advancement of Pure Research (ZWO).

the packet has to be stored (queued) temporarily in the memory of the intermediate nodes on the path of this packet. For this purpose each processor has an (integer) number of buffers in which packets can be stored. We assume each node has b buffers, where b is a fixed constant. Each packet p has an (integer) size, denoted by s(p), which is the number of buffers the packet needs when stored at a node. The maximum size of a packet is assumed to be q, so for each packet p one has $1 \leq s(p) \leq q$. The sum of the sizes of the packets stored at a certain node on a certain moment can never exceed b. We also assume it is possible to store a packet p at node v, if s(p) plus the sum of the sizes of the packets currently stored at v does not exceed b. (Later we will forbid the acceptance of packets p at nodes v in some cases, even when it is theoretically possible to store p at v.) Suppose the buffers of a node v are numbered 1,...,b. In some cases it is necessary to store a packet in buffers, that are not consequtively numbered, or to reallocate one or more packets internally in the memory of the processors. For an example see fig. 1.1. Let b=6, $q \geq 2$. At a node v packets with sizes 2, 1 and 2 (in this order) are stored (fig. 1.1.a.). The packet with size 1 leaves v, the buffer where it was stored is "free" again (fig. 1.1.b.). To store another packet with size 2, we can either reallocate the second packet with size 2 (fig. 1.1.c.) or we have to store the packet at non-consequtive buffers (fig. 1.1.d.). We will further ignore this problem.

We consider three types of "moves", made in the network:

i) Generation of a packet: a node v generates a packet p and places it in s(p) of its empty buffers.

ii) Passing of a packet: a packet p is passed from a node v to a node w ($(v,w) \in E$) in accordance to the followed routing strategy; the buffers in v where the packet was stored become empty and the packet is placed in s(p) empty buffers of w.

iii) Consumption of a packet: a packet p that is stored at the buffers of its destination node v is removed from the buffers of v; the s(p) buffers where p was stored become empty.

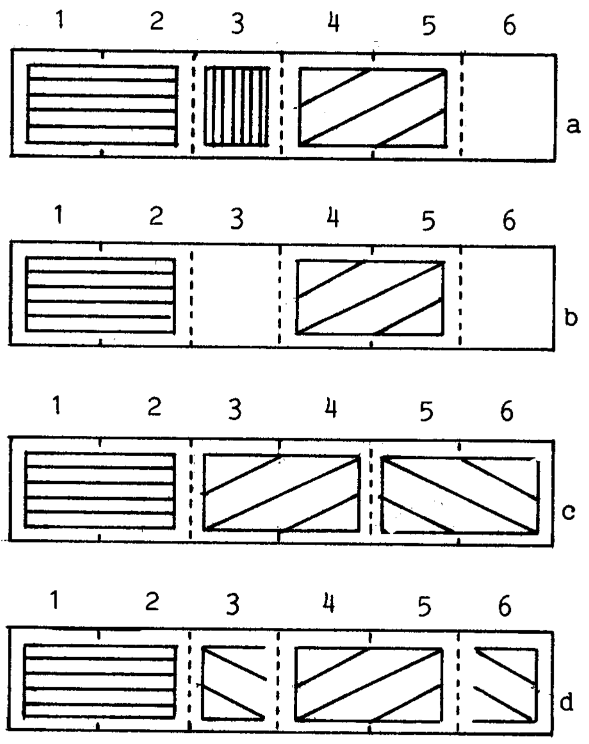

fig. 1.1.

For our analysis we assume that moves of the network take one unit of time each: each moment t is either before a specific move or after this move, never "during" the move. This is no real limitation of the model, buf facilitates the analysis.

A (distributed) algorithm that permits some moves of the network and forbids others is called a controller. An important and very desirable property of controllers is that they prevent deadlock. A deadlock occurs if there is a situation when there are packets which will never arrive at their destination, no matter what sequence of moves is performed by the network. A (classic) example of a network with deadlock is given in fig. 1.2. Assume the controller permits all moves that do not let the sum of the sizes of the packets stored at v exceed b. If in v_1 b packets are created with size 1 and destination

v_2, in v_2 b packets are created with size 1 and destination v_3 and in v_3 b packets are created with size 1 and destination v_1, then the network cannot make another move, so there is a deadlock.

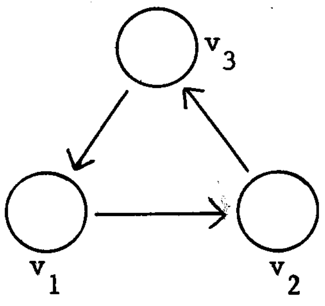

fig. 1.2. A network with a controller that does not prevent deadlock.

Definition. (Toueg and Ullman [4]). A controller is deadlock-free for a given network, if it does not permit the network to enter a state in which one or more packets can never make a move provided that no additional packets are generated.

Toueg and Ullman [4] noted that the condition that no additional packets are generated is essential to exclude certain strange controllers, where a packets can only move if some other packet is generated, and that packet can only move if a third packet is generated, and so on.

We will only consider local controllers: controllers that use only information about the local state of a node and the state of a packet, but do not use global information. Global controllers have an important drawback: if the size of the network becomes large it can cost much additional work and messages to gain global knowledge over the state of the network.

Several deadlock-free controllers, that use only local information are known. (See e.g. [1], [2], [3] and [4].) Each of these controllers assumes that packets have a unit size, i.e. $q=1$. Toueg and Ullman [4] proposed 4 controllers, that use only the state of a packet p, and the state of the packets stored at node v, to decide whether v can accept p or not, called: forward-count, backward-count, forward-state and backward-state, and proved each to be deadlock-free. We generalize these results in three ways:

- we allow packets to have a variable size,
- we show that "forward-count" and "backward-count" are special cases of the more general notion of "count-down", and thus obtain a uniform theory of deadlock-free controllers of this type, and
- we show that every controller that is "in between" a count-controller and the corresponding state-controller is also deadlock-free.

(For more precise definitions of the notions used here, see section 2.) This paper is organized as follows. In section 2 we give some basic definitions. In section 3 we present a large class of local deadlock-free controllers. In section 4 we show that some of the controllers in this class are optimal in the class of all deadlock-free controllers using the same local information. Some final comments are made in section 5.

2. Definitions.

Definition. A count-down function Φ is a partial function that maps a triple, consisting of a packet p, a node v and time t to an integer, such that:

i) If p is stored at v on moment t, or v is the next node on the path of p on moment t, then $\Phi(p,v,t)$ is defined.

ii) If $\Phi(p,v,t)$ is defined then $\Phi(p,v,t) \in \{o,1,\dots,k\}$.

iii) If $\Phi(p,v,t)$ and $\Phi(p,v,t')$ are defined and $t<t'$, then $\Phi(p,v,t) \geq \Phi(p,v,t')$.

iv) If $\Phi(p,v,t)$ and $\Phi(p,v',t)$ are defined, and v is before v' on the path of p from its source to its destination, then $\Phi(p,v,t) > \Phi(p,v',t)$.

For a count-down function Φ, it means that every time a packet p is passed to a next node on its path, the Φ-value of p drops by at least one. So, for a packet p stored at v at time t, $\Phi(p,v,t)$ is an upperbound on the number of steps p has still to go to arrive at its destination node. If p is stored at v at time t and $\Phi(p,v,t)=o$, then v is the destination of p.

Important examples of count-down functions are the following.

i) Assume that a fixed routing strategy is used, i.e. for each source destination pair w_o,w_1, there is a unique path from w_o to w_1 that is followed by all packets that are send from w_o to w_1. For a packet p and a node v on the path of p, and for all t, we let $F(p,v,t)$ denote the distance from v to the destination of p, i.e. the number of message passing steps p has to take to arrive at its destination. (F stands for "forward".)

ii) For a packet p and a node v on the path of p, let $\tilde{B}(p,v)$ denote the distance from the source of p to v, i.e. the number of message passing steps p has to take from its source to arrive at v. (Note: $\tilde{B}$ is not a count-down function.) Now let for all times t $B(p,v,t) = k-\tilde{B}(p,v)$. (B stands for "backward".)

It is easy to verify that F and B are count-down functions. Note that we always can choose B as a count-down function. (This basically follows from the assumption that each packet needs to traverse at most k links to arrive at its destination.)

Another example of a count-down function is the following. Basically B is used as the count-down function. However, as an addition, in each node v, after a packet p is stored in the buffers of v, it is tested whether one of the nodes adjacent to v is the destination of p. Is this the case, then immediately the value $\Phi(p,v,t)$ drops to 1. (Note that the "Φ-value" of p can change, even if p is not moved.)

$\Phi(p,v,t)$ can be seen as an upperbound on the number of message passing steps p has to take from v to arrive at its destination; the closer $\Phi(p,v,t)$ is to the exact number of steps p has to take, the less restrictive our resulting controllers will be. In this sense F is the "best" count-down function, and B is the "worst":

<u>Lemma 2.1</u>. Let p be a packet stored at v at time t, or let v be the next node on the path of p. Let Φ be a count-down function.

i) $\Phi(p,v,t) \leq B(p,v,t)$.

ii) Assume a fixed routing strategy is used. $\Phi(p,v,t) \geq F(p,v,t)$.

<u>Proof</u>.

i) v is the node of index $k-B(p,v,t)+1$ on the path of p. Let $v_o,v_1,\dots,v_{k-B(p,v,t)} = v$ be the first $k-B(p,v,t)+1$ nodes on this path, and let times $t_o<t_1<_2<\dots<t_{k-B(p,v,t)+1} = t$ be given such that $\Phi(p,v_i,t_i)$ is defined, for all i, $o\leq i\leq k-B(p,v,t)$. Then $k\geq\Phi(p,v_o,t_o) > \Phi(p,v_1,t_1) >\dots> \Phi(p,v_{k-B(p,v,t)}, t_{k-B(p,v,t)}) = \Phi(p,v,t)$, hence $\Phi(p,v,t) \leq B(p,v,t)$.

ii) Similar. (If not a fixed routing strategy is used, then F is not defined.) □

Now we define the <u>state</u> of packets and of nodes. Let Φ be a count-down function. The state of packet p, stored at node v at time t can be described by the pair $(s(p),\Phi(p,v,t)) \in \{1,\dots,q\} \times \{o,\dots,k\}$. The state of a node v at time t is formed by the states of the packets that are stored at v at time t. We can represent this state by a q by (k+1) matrix J, with J(s,d) denoting the number of packets stored at v with state (s,d), ($1\leq s\leq q$, $o\leq d\leq k$). J is called the statematrix of v. We use the following notions, derived from the statematrix J:

<u>Definition</u>. Let J be the statematrix of a processor v at certain time t.

i) The <u>statevector</u> of J is the vector $\vec{j} = \vec{j}(J)=\langle j_o,\dots,j_{kq}\rangle$, with $j_i = \Sigma\, J(s,d).s$, where the sum is taken over all pairs (s,d), with $s.d = i$, $1\leq s\leq q$, $o\leq d\leq k$, i.e. j_i is the number of buffers used by packets with state (s,d) with $s.d = i$.

ii) The number of used buffers at v is denoted by $n = \sum_{r=o}^{kq} j_r = \sum_{r_1=1}^{q} \sum_{r_2=o}^{k} (J(r_1,r_2).r_1)$.

iii) The number of free buffers at v is denoted by $m = b-n$.

The controllers we consider in this paper will allways allow consumption moves, but will disallow some generation and message passing moves. We consider controllers where the decision to accept a packet p at a node v at time t only depends on the pair $(s(p),\Phi(p,v,t))$, the statematrix of v, and on the global constants q, k and b. A controller, using a count-down function Φ, and constants q (for maximum packet size), k (for maximum Φ-values) and b (for buffersize) is called a uniform (Φ,q,k,b)-controller. A network with count-down function Φ, maximum packet size q, maximum Φ-value on the network k and buffersize b is called a (Φ,q,k,b)-network. A uniform (Φ,q,k,b)-controller can be used for every (Φ,q,k,b)-network.

Formally, a uniform (Φ,q,k,b)-controller can be described as a subset $S \subseteq \{(J,(s,d)) \mid 1 \leq s \leq q,\ o \leq d \leq k,$ J is a q by k+1 matrix with non-negative values and $\sum_{r_1=1}^{q} \sum_{r_2=o}^{k} J(r_1,r_2).r_1 \leq b-s\}$. It means that a packet p can be accepted at node v at time t, with J is the statematrix of v, if and only if $(J,(s(p),\Phi(p,v,t))) \in S$.

If we do not use the statematrix J, but statevectors $\vec{j}$ or numbers n or m, we will describe the controllers with sets of pairs $(\vec{j},(s,d))$, $(n,(s,d))$ or $(m,(s,d))$. Controllers that do not use J or $\vec{j}$, but only n or m are called "count"-controllers, the others are called "state"-controllers. In the case that q=1 we also use pairs $(\vec{j},d)$, (n,d) and (m,d).

3. <u>Uniform local deadlock-free controllers</u>.

We first recall some results of Toueg and Ullman [4]. Consider the following 4 controllers. In each case we have q=1 and $b \geq (k+1)q = k+1$. (For i) and ii) assume that a fixed routing strategy is used.)

i) A node v with m free buffers accepts a packet p at time t iff $(m,F(p,v,t)) \in FC(b,k) = \{(m,j) \mid (j<m)$ and $(o \leq j \leq k)$ and $(1 \leq m \leq b)\}$. (FC stands for "forward count".)

ii) A node v with statevector $\vec{j}$ (with respect to count-down function F) accepts a packet p at time t iff $(\vec{j},F(p,v,t)) \in FS(b,k)$ $= \{(\vec{j},j) \mid (\forall i,\ o \leq i \leq j,\ i < b - \sum_{r=i}^{k} j_r)$ and $(o \leq j \leq k)$ and $(o \leq \sum_{r=o}^{k} j_r$

$\leq b-1)\}$. (FS stands for "forward state".)

iii) A node v with n used buffers accepts a packet p at time t, iff $(n,B(p,v,t)) \in BC(b,k) = \{(n,i) \mid (i \geq n-b+(k+1))$ and $(o \leq i \leq k)$ and $(o \leq n \leq b-1)\}$. (BC stands for "backward count".)

iv) Let v be a node with statevector $\vec{j}$, with respect to count-down function B. Write $\vec{i} = \langle i_o,\ldots,i_k \rangle$ with $i_r = j_{k-r}$ $(o \leq r \leq k)$, so i_r denotes the number of packets, stored at v, that have made r steps so far, i.e. the distance from the source of these packets to v is r. Recall that $\tilde{B}(p,v)$ denotes the distance from the source of p to v. v accepts a packet p at time t, iff $(\vec{i},\tilde{B}(p,v)) \in BS(b,k) = \{(\vec{i},i) \mid (\forall j,\ i \leq j \leq k,\ j \geq \sum_{r=o}^{j} i_r -b+(k+1))$ and $(o \leq i \leq k)$ and $(o \leq \sum_{r=o}^{k} i_r \leq b-1)\}$. (BS stand for "backward state".)

Theorem 3.1. [4]

i) For $b \geq k+1$, $FC(b,k)$ and $FS(b,k)$ are deadlock-free uniform $(F,1,k,b)$-controllers.

ii) For $b \geq k+1$, $BC(b,k)$ and $BS(b,k)$ are deadlock-free uniform $(B,1,k,b)$-controllers.

A generalization of the controllers FC(b,k) and BC(b,k) is the following class of count-controllers:

- Let Φ be some count-down function. A node v accepts a packet p at time t, iff $(m,(s(p),\Phi(p,v,t))) \in \Phi CV(q,k,b) = \{(m,(s,d)) \mid ((d+1)s \leq m)$ and $(1 \leq s \leq q)$ and $(o \leq d \leq k)$ and $(1 \leq m \leq b)\}$. $\Phi CV(q,k,b)$ is a uniform (Φ,q,k,b)-controller.

A generalization of the controllers FS(b,k) and BS(b,k) is the following class of state-controllers:

- Again Φ is some count-down function. A node v, with state vector $\vec{j}$ (with respect to Φ) accepts a packet p at time t, iff $(\vec{j},(s(p),\Phi(p,v,t))) \in \Phi SV(q,k,b) = \{(\vec{j},(s,d)) \mid (\forall i,\ o \leq i \leq ds,\ i+s \leq b - \sum_{r=i}^{kq} j_r)$ and $(1 \leq s \leq q)$ and $(o \leq d \leq k)$ and $(o \leq \sum_{r=o}^{kq} j_r \leq b-s)\}$. $\Phi SV(q,k,b)$ is a uniform (Φ,q,k,b)-controller.

uniform (Φ,q,k,b)-controller.

If $\Phi=F$, then we write $\Phi CV(q,k,b)=FCV(q,k,b)$ and $\Phi SV(q,k,b) = FSV(q,k,b)$. Likewise for $\Phi=B$, etc. If q,k and b are clear from the context, then we write ΦCV, and ΦSV.

For uniform (Φ,q,k,b) controllers S and T we write $S \subseteq T$, iff every move that is allowed by S, is also allowed by B. If $S\subseteq T$ and $T \subseteq S$ then we write $S\equiv T$. If $S \subseteq T$ and not $S\equiv T$, then we write $S \subset T$.

Lemma 3.2. Let $q=1$, and $b > k+1$.

i) $FC(b,k) \equiv FCV(1,k,b)$.

ii) $BC(b,k) \equiv BCV(1,k,b)$.

iii) $FS(b,k) \equiv FSV(1,k,b)$.

iv) $BS(b,k) \equiv BSV(1,k,b)$.

Proof.

We only give a proof of iv), the other results follow directly from the definitions. Let p be a packet that wants to be accepted by node v at time t. Let $\vec{j}$ be the statevector of p (with respect to count-down function B), and let $\vec{i} = \langle i_o,\ldots,i_k\rangle$ be given by $i_r = b_{k-r}$ $(o\le r\le k)$. Further write $s = s(p)$, $i=\tilde{B}(p,v) = k-B(p,v,,t)$. Now we have:

BS(b,k) lets v accept p

$\Leftrightarrow(\forall j,\ i\le j\le k,\ j\ge \sum_{r=o}^{j} i_r - b+(k+1))$ and $(o\le i\le k)$ and $(o\le \sum_{r=o}^{k} i_r \le b-1)$

$\Leftrightarrow(\forall j,\ i\le j\le k,\ j\ge \sum_{r=k-j}^{k} j_r - b+(k+1))$ and $(o\le i\le k)$ and $(o\le \sum_{r=o}^{k} j_r \le b-1)$

$\Leftrightarrow(\forall j,\ o\le j\le k-i,\ k-j\ge \sum_{r=j}^{k} j_r - b+(k+1))$ and $(o\le k-i\le k)$ and $(o\le \sum_{r=o}^{k} j_r \le b-1)$

$\Leftrightarrow(\forall j,\ o\le j\le B(p,v,t),\ j+1\le b-\sum_{r=j}^{k} j_r)$ and $(o\le B(p,v,t)\le k)$ and $(o\le \sum_{r=o}^{k} j_r \le b-1)$

$\Leftrightarrow$BSV (1,k,b) lets v accept p. □

Lemma 3.3. Let q,k,b be given and let Φ_1,Φ_2 be count-down functions, such that $\forall p,v,t$, if $\Phi_1(p,v,t)$ is defined and $\Phi_2(p,v,t)$ is defined then $\Phi_1(p,v,t) \geq \Phi_2(p,v,t)$. Then

i) $\Phi_1 CV(q,k,b) \subseteq \Phi_2 CV(q,k,b)$.

ii) $\Phi_1 SV(q,k,b) \subseteq \Phi_2 SV(q,k,b)$.

Proof.

i) This follows directly from the definition of ΦCV.

ii) Let v be a node. Let the statevector of v at time t with respect to Φ_1 be $\vec{j}^1 = \langle j_o^1, j_1^1, \ldots, j_{kq}^1 \rangle$, and let the statevector of v at time t with respect to Φ_2 be $\vec{j}^2 = \langle j_o^2, j_1^2, \ldots, j_{kq}^2 \rangle$. Because we have for each packet p^1, stored in v at time t, that $\Phi_1(p^1,v,t) \geq \Phi_2(p^1,v,t)$, it follows that $\sum_{r=i}^{kq} j_r^1 \geq \sum_{r=i}^{kq} j_r^2$, for every i, $o \leq i \leq kq$. Now suppose $\Phi_1 CV$ lets v accept packet p at time t. Then $\forall i, o \leq i \leq \Phi_1(p,v,t).s(p)$, $i+s(p) \leq b - \sum_{r=i}^{kq} j_r^1$, so $\forall i, o \leq i \leq \Phi_2(p,v,t).s(p)$, $i+s(p) \leq b - \sum_{r=i}^{kq} j_r^1 \leq b - \sum_{r=i}^{kq} j_r^2$, hence $\Phi_2 CV$ lets v accept p at time t. □

Lemma 3.4. For all $q \geq 1$, $k \geq 1$, $b \geq (k+1)q$, and count-down functions Φ, $\Phi CV(q,k,b) \subset \Phi SV(q,k,b)$.

Proof.

We first show that $\Phi CV \subseteq \Phi SV$. Suppose ΦCV lets a node v with statevector $\vec{j}$ at time t accept a packet p. Write $s=s(p)$, and $d=\Phi(p,v,t)$. Now $(d+1)s \leq b - \sum_{j=o}^{kq} j_r$. Hence, for all i, $o \leq i \leq ds$, $i+s \leq ds+s \leq b - \sum_{j=o}^{kq} j_r \leq b - \sum_{j=i}^{kq} j_r$, so ΦSV lets v accept p, at time t.

Next we show there are moves allowed by ΦSV, that are not allowed by ΦCV. Consider a node v with an empty buffer. Let v accept successively $b(v)-1$ packets with $\Phi(p,v,t)=o$ and $s(p)=1$ (this is possible with ΦCV and with ΦSV). Now try to let v accept a packet p^* with $\Phi(p^*,v,t)=1$ and $s(p^*)=1$. It easily follows that ΦSV allows this move, but ΦCV does not allow this move. (There are many other examples.) □

We now give our main theorem. The proof of this theorem is similar to the proof in [4], that FS is deadlock-free. The introduction of the notion of count-down functions allows us to treat the "forward" and "backward" controllers as special cases of a more general notion.

Theorem 3.5. Let $q \geq 1$, $k \geq 1$, $b \geq (k+1)q$, and Φ a count-down function. Let S be a uniform (Φ,q,k,b)-controller with $\Phi CV(q,k,b) \subseteq S \subseteq \Phi SV(q,k,b)$. Then S is deadlock-free.

Proof.

Suppose S is not deadlock-free. Then consider a (Φ,q,k,b)-network G and suppose the network reaches a state where one or more packets are deadlocked. By making every move that is possible, we can reach a state, where every packet is deadlocked. So no packet in the network can move at a certain time t, and there is at least one packet p in the network. Let p_1 be such a packet; let v_1 be the node where p_1 is stored, and let $d_1 = \Phi(p_1,v_1,t)$, $s_1 = s(p_1)$. It is clear that v_1 is not the destination of p_1, else v_1 can consume p_1. Let v_2 be the next node on the path of p_2 to its destination, after v_1, and let $\vec{j} = \{j_o,\ldots,j_{kp}\}$ be the statevector of v_2. Now note that $\sum_{r=o}^{kq} j_r \neq o$, else p_1 is accepted in v_2. So there is at least one other packet in v_2, this packet can also not move. Let p_2 be a packet in v_2, such that $\Phi(p_2,s_2,t)s(p_2)$ is minimal over all packets in v_2:

$$\Phi(p_2,v_2,t).s(p_2) = \min \{r \mid j_r > o\}.$$

Let $d_2 = \Phi(p_2,v_2,t)$, $s_2 = s(p_2)$. The statevector of v_2 can be written as $\vec{j} = \langle o,\ldots,o,j_{d_2s_2},\ldots,j_{kq}\rangle$. We have that $d_2 \neq o$, else p_2 can consume v_2. Now we claim that $d_2s_2 < d_1s_1$.

Suppose p_s is the last packet that is accepted by v_2, let t_s be the moment on which p_s was accepted by v_2. (Note that at time t_s p_s is not yet stored at v_2.) Let $\vec{j}^s$ be the statevector of v_2 at time t_s. We write $d_s = \Phi(p_s,v_2,t_s)$, $s_s = s(p_s)$. We have that $d_ss_s \geq \Phi(p_s,v_2,t)s_s \geq d_2s_2$. Note that packets stored at v_2 at t_s can have been moved out of v_2 (either by a consumption move or a passing move). Also for some packets, stored at v_2 at time t_s and time t, it can be that $\Phi(p_s,v_2,t_s) > \Phi(p_s,v_2,t)$. For all these packets $\Phi(p_s,v_2,t_s) \geq \Phi(p_s,v_2,t)$. The only packet that is newly stored in the buffers of v_2 is p_2. Therefore one has:

$$\forall i,\ o \leq i \leq d_ss_s\ ,\ \sum_{r=i}^{kq} j_r^s \geq \sum_{r=i}^{kq} j_r - s_s$$

$$\forall i,\ d_s s_s < i \leq kq,\ \sum_{r=i}^{kq} j_r^s \geq \sum_{r=i}^{kq} j_r.$$

S lets a node with statevector $\vec{j}^s$ accept a packet with state (s_s, d_s), and $S \subseteq \Phi SV(q,k,b)$, hence this move is also allowed by $\Phi SV(q,k,b)$ and therefore one has:

$$\forall i,\ o \leq i \leq d_s s_s,\ i + s_s \leq b - \sum_{r=i}^{kq} j_r^s.$$

Take $i = d_2 s_2$ and one gets:

$$d_2 s_2 + s_s \leq b - \sum_{r=d_2 s_2}^{kq} j_r^s \leq b - \sum_{r=d_2 s_2}^{kq} j_r + s_s ,$$

hence

$$d_2 s_2 \leq b - \sum_{r=d_2 s_2}^{kq} j_r.$$

We have also that S does not let v_2 accept p_1 at time t. Because $\Phi CV(q,k,b) \subseteq S$, this move is also not allowed by $\Phi CV(q,k,b)$. So we have that

$$d_1 s_1 \geq (\Phi(p_1, v_2, t) + 1)\ s_1 > b - \sum_{r=o}^{kq} j_r$$ (= the number of free buffers in v_2 at time t.)

Now suppose $d_2 s_s \geq d_1 s_1$. Then $b - \sum_{r=d_2 s_2}^{kq} j_r = b - \sum_{r=o}^{kq} j_r < d_1 s_1 \leq d_2 s_2 \leq b - \sum_{r=d_2 s_2}^{kq} j_r$. Contradiction. Hence $d_2 s_2 < d_1 s_1$.

The argument now can be repeated with p_2 instead of p_1, and so on. In this way we obtain packets $p_1, p_2, p_3, \ldots$, stored at nodes $v_1, v_2, v_3, \ldots$ (these nodes are not necessarily all different), with $\Phi(p_1, v_1, t)s(p_1) > \Phi(p_2, v_2, t)s(p_2) > \Phi(p_3, v_3, t)s(p_3) > \ldots$etc. This contradicts the fact that all values $\Phi(p_i, s_i, t)$ and $s(p_i)$ are non-negative. Hence S must be deadlock-free. □

<u>Corollary 3.6</u>. Let $q \geq 1$, $k \geq 1$, $b \geq (k+1)q$, and let Φ be a count-down function. $\Phi CV(q,k,b)$ and $\Phi SV(q,k,b)$ are local, uniform (Φ,q,k,b)-controllers, that are deadlock-free.

4. Optimality of ΦSV.

Definition. Let q,k,b and a count-down function Φ be given.

i) Φ is strict, if it is possible to use Φ as a count-down function in all networks G, for which we do not have to send messages between nodes that have a distance of more than k.

ii) Φ is stable, if for all packets p, nodes v and times t,t', if $\Phi(p,v,t)$ and $\Phi(p,v,t')$ are defined, then $\Phi(p,v,t) = \Phi(p,v,t')$.

If Φ is a strict count-down function, then we can use it in all networks where we want to send messages between each pair of nodes with a distance k between each other. An example of a count-down function, that is not strict is the function Φ, with, if $F(p,v,t)$ is defined, then $\Phi(p,v,t) = 2F(p,v,t)$. Note that, for a strict count-down function Φ, a packet p, with the distance between the source of p and the destination of p is k, and a node v on the path of p, one has $\Phi(p,v,t) = B(p,v,t)$, and if F exists (i.e. a fixed routing strategy is used), then $\Phi(p,v,t) = F(p,v,t)$.

In this section we will consider strict and stable count-down functions only. Important examples of strict and stable count-down functions are F and B. The results in this section are mainly rather straightforward generalizations of the work of Toueg and Ullman [4, p.598-607]. We generalize these results in two ways:

- we allow packets to have a variable size
- the results are valid for every strict and stable count-down function. (Toueg and Ullman only considered controllers, depending on F or B.)

Let integers $q \geq 1$, $k \geq 1$, $b \geq 1$ and strict, stable count-down function Φ be given. Suppose S is a local, uniform (Φ,q,k,b)-controller. For statematrices J_o and J_1 we write $J_o \underset{S}{\vdash} J_1$, iff and only if there is exactly one pair $(s,d) \in \{1,\dots,q\} \times \{o,\dots,k\}$ with $J_o(s,d) \neq J_1(s,d)$, and

i) for this pair (s,d) one has $J_o(s,d)-1 = J_1(s,d) \geq o$ ("a packet with state (s,d) has left the node") or

ii) for this pair (s,d) one has $J_o(s,d)+1 = J_1(s,d)$ and $\forall t$ $(J_o,(s,d,t)) \in S$, i.e. S allows a node with state matrix J_o to accept a packet with state (s,d).

The transitive closure of the relation $\underset{S}{\vdash}$ is denoted by $\underset{S}{\vdash^*}$: $J_o \underset{S}{\vdash^*} J_1$, if there are $J^o=J_o, J^1, J^2, \ldots, J^i=J_1$ with $J^o \underset{S}{\vdash} J^1$, $J^1 \underset{S}{\vdash} J^2, \ldots, J^{i-1} \underset{S}{\vdash} J^i$. Let Q be the statematrix consisting of only zero's: $Q(s,d) = o$ for all s,d, $1 \leq s \leq q$, $o \leq d \leq k$. If $Q \underset{S}{\vdash^*} J$, then we write $\underset{S}{\vdash^*} J$ (J is "syntactical reachable"). Conversely, we write $\underset{S}{\models^*} J$ (J is "network reachable"), iff there exists a network G, such that, starting from the state where all buffers of all nodes are empty, we can make a series of moves, allowed by S, such that after these moves there is a node with state J.

<u>Lemma 4.1</u>. Let Φ be a strict and stable count-down function, let $q \geq 1$, $k \geq 1$, $b \geq 1$. Let S be a local, uniform (Φ,q,k,b)-controller. Then for each statematrix J:

$$\underset{S}{\vdash^*} J \Leftrightarrow \underset{S}{\models^*} J.$$

<u>Proof</u>.

Suppose $\underset{S}{\vdash^*} J$, so there are statematrices $J^1, J^2, \ldots, J^i$, with $Q \underset{S}{\vdash} J^1$, $J^1 \underset{S}{\vdash} J^2, \ldots, J^{i-1} \underset{S}{\vdash} J^i$. With induction to j we will show that in the unidirectional ring with $k+1$ nodes R_{k+1} (see fig. 4.1.) there is a series of moves, allowed by S, starting from the state where all buffers are empty, such that after the moves v_o has statematrix J^j, and all other nodes have statematrix Q. Write $J^o = Q$. It is clear, that the induction hypothesis holds for $j=o$. Now suppose the induction hypothesis holds for certain j. Then we can make a series of moves, starting from the state where all buffers of all nodes are empty, resulting in the state where v_o has statematrix J^j, and all buffers of all other nodes empty. We have that $J^i \underset{S}{\vdash} J^{j+1}$. If there is a pair (s,d) with $J^j(s,d) - 1 = J^{j+1}(s,d) \geq o$, then we can move a packet with state (s,d) to its destination and consume it there. If there is a pair (s,d) with $J^i(s,d) + 1 = J^{j+1}(s,d)$, and $\forall t, (J^j,(s,d,t)) \in S$, then create a packet p with size s in v_{d+1}, with destination v_d. This packet can be moved to v_o, and, because $(J^j,(s,d)) \in S$, it can be accepted in v_o. (Note that $\forall t\ \Phi(v_o,p,t) = B(v_o,p,t) = d$.) Hence the induction hypothesis holds also for j+1. We conclude that $\underset{S}{\models^*} J$.

Now suppose that $\underset{S}{\vdash^*} J$. From the stableness of Φ it follows that the state of a node v can only change if a packet is accepted in v (by a passing move or a generation move), or if a packet leaves v (by a passing move or a consumption move). Both types of changes correspond to a statetransition $J^o \underset{S}{\vdash} J^1$. With induction one can conclude $\underset{S}{\vdash^*} J$. □

States J with not$(\underset{S}{\vdash} J)$ ($\Leftrightarrow$not$(\underset{S}{\models} J)$) are called unreachable states. Without loss of generality, we can assume that considered controllers do not allow moves from unreachable states. We are now ready to give the main result in this section.

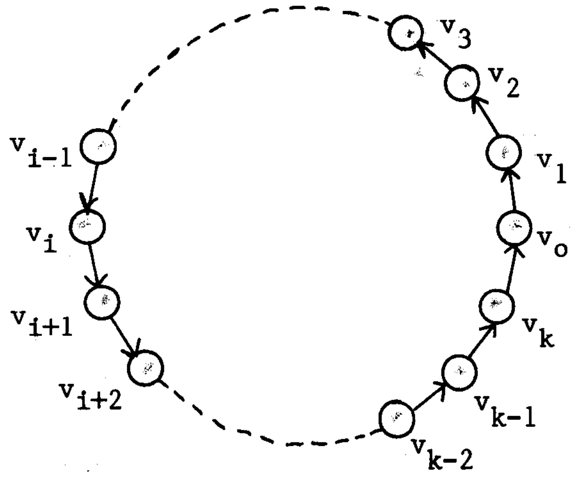

fig. 4.1. The unidirectional ring R_{k+1}.

Theorem 4.2. Let $q \geq 1$, $k \geq 1$, $b \geq 1$, let Φ be a strict and stable count-down function. Let S be a local, uniform (Φ,q,k,b)-controller, that does not allow moves from unreachable states.

i) If $b < (k+1)q$, then S is not deadlock-free.

ii) If $b \geq (k+1)q$ and S is deadlock-free, then $S \subseteq \Phi SV(q,k,b)$.

Proof.

Let q, k, b, Φ and S be given. Suppose that S is deadlock-free and $b<(k+1)q$ or not($S \subseteq \Phi SV(q,k,b)$). We first claim there is a pair (s,d) $\in \{1,\ldots,q\} \times \{o,\ldots,k\}$ and a statematrix J, such that $(J,(s,d)) \in S$ and for $\vec{j} = \vec{j}(J)$:

$$\exists\, i_o,\ o \leq i_o \leq sd,\ i_o+s > b- \sum_{r=i_o}^{kq} j_r.$$

If $b<(k+1)q$, then we can choose $\vec{j} = \langle o,\ldots,o\rangle$, d=k, s=q and $i_o = kq$. If not($S \subseteq \Phi SV(q,k,b)$), then the result follows directly from the definition of $\Phi SV(q,k,b)$. When v acceps a packet with size s, v must have at least s free buffers, so $i_o+s > b- \sum_{r=i_o}^{kq} j_r \geq b- \sum_{r=o}^{kq} j_r \geq s$, hence $i_o \geq 1$.

Consider all pairs (s,d), such that there exists a statematrix J, with $(J,(s,d)) \in S$, and for $\vec{j} = \vec{j}(J)$ one has that

$$\exists\, i_o,\ o \leq i_o \leq sd,\ i_o+s > b- \sum_{r=i_o}^{kq} j_r.$$

Let (s_o,d_o) be such a pair, such that s_od_o is minimal over all these pairs. Let i_o be given, such that $o \leq i_o \leq s_od_o$, $i_o+s > b- \sum_{r=i_o}^{kq} j_r$, and let J_o be the corresponding statematrix. As noted before, $i_o \geq 1$, so $s_od_o \geq 1$, hence $d_o \geq 1$. Consider the graph G, given in fig. 4.2.

In the network G we will only send messages between nodes that have a distance k, i.e. we send messages from nodes v_i to nodes v_{i-1}, nodes v_i to nodes w_{i-2}, etc. We now give a series of moves, permitted by S, starting from the state where all buffers are empty, such that a deadlock will occur in G. We only describe the moves made by the nodes v_i $(o \leq i \leq k)$; the same moves are made by the nodes w_i $(o \leq i \leq k)$.

1. As in the proof of lemma 4.1. we can reach state J_o in v_o, with messages with source and destination in $\{v_i \mid o \leq i \leq k\}$. Let $\vec{j} = \vec{j}(J_o)$.
2. Generate a packet p with size s_o, in node $v_{(d_o+1)\bmod(k+1)}$ with

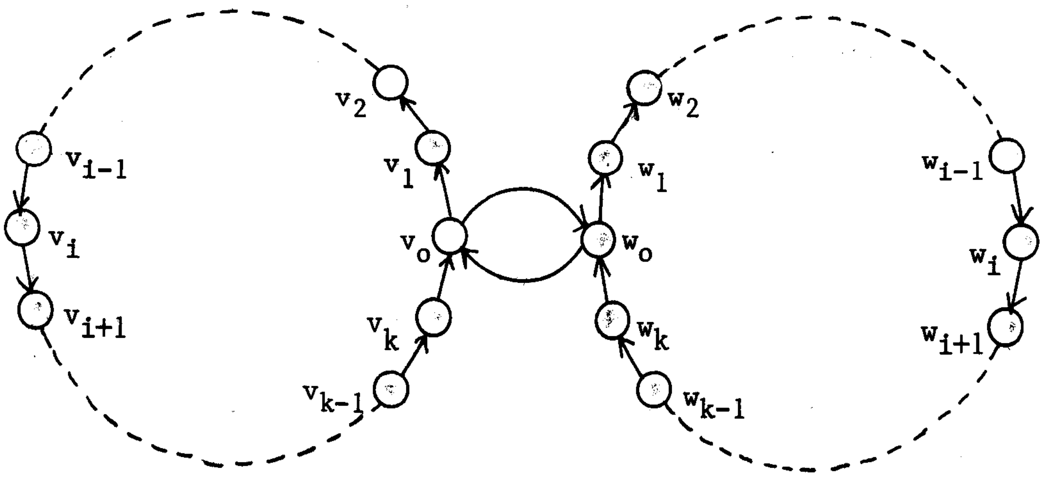

fig. 4.2.

destination $w_{(d_o-1)}$. Note that $\forall t$, $\Phi(p,v_o,t) = B(p,v_o,t) = d_o$. Pass p along the successive nodes on the ring $\{v_i | o \le i \le k\}$, and accept it in v_o (this is possible, because $(J_o,(s_o,d_o)) \in S$). Let J_1 be the statematrix of v_o, just after p is placed in the buffers of v_o, and write $\vec{j}^1 = \vec{j}(J^1)$. We have that

$j_r^1 = j_r^o$, if $r \ne s_o d_o$, and
$j_{s_o d_o}^1 = j_{s_o d_o}^o + s_o$.

3. Let all packets p, in v_o, with $\Phi(p,v_o,t).s(p) \le i_o$ be passed to their destination, and consumed there. Let the resulting statematrix of v_o be J_2, and write $\vec{j}^2 = \vec{j}(J_2)$. Now we have
$j_r^2 = o$, if $o \le r \le i_o - 1$, and
$j_r^2 = j_r^1 = j_r^o$, if $i_o \le r \le kq$ and $r \ne s_o d_o$, and
$j_{s_o d_o}^2 = j_{s_o d_o}^1 = j_{s_o d_o}^o + s_o$.
4. The destination of the packets that still are in v_o is changed, such that the next node on their path is w_o. (So a packet p with $\Phi(p,v_o,t) = d$, now has destination w_{d-1}.) Note that for each packet p in v_o one has $\forall t$ $B(p,v_o,t) = F(p,v_o,t) = \Phi(p,v_o,t) \ge i_o \ge 1$.
5. As long as it is possible to acept in v_o packets p with $s(p).\Phi(p,v_o,t) \ge s_o d_o$ (with regard to S), such packets p are generated in $v_{(\Phi(p,v_o,t)+1) \bmod (k+1)}$ with destination $w_{\Phi(p,v_o,t)-1}$. The packets are passed along the successive nodes on the ring $\{v_i | o \le i \le k\}$, and are accepted by v_o. We stop iff this is no longer

possible, i.e. v_o reaches a state(matrix) J_3, with for all s,d; $o\leq d\leq k$, $1\leq s\leq q$, $sd\geq s_o d_o$, $(J_3,(s,d)) \notin S$. Write $\vec{j}^3 = \vec{j}(J_3)$. We have that

$j_r^3 = o$, iff $o\leq r\leq i_o-1$, and
$j_r^3\geq j_r^2=j_r^o$, iff $i_o\leq r\leq kq$ and $r\neq s_o d_o$, and
$j_{s_o d_o}^3 \geq j_{s_o d_o}^2 = j_{s_o d_o}^o + s_o$.

Now G is deadlocked. There is at least one packet in v_o ($j_{s_o d_o}^3 \geq s_o$), and every packet in v_o is directed towards w_o. The state matrix of w_o is also J_3. We now claim $\forall$ s,d, $1\leq s\leq q$, $o\leq d\leq k$, $sd\geq s_o d_o$: $(J_3,(s,d-1)) \notin S$. Suppose there are s,d, $1\leq s\leq q$, $o\leq d\leq k$, $sd\geq s_o d_o$ and $(J_3,(s,d-1)) \in S$. First note that $(d-1)s<s_o d_o$ (because of the construction of J_3). Further note that $b-\sum_{r=i_o-s}^{kq} j_r^3 \leq b-(j_{i_o}^o+\ldots+(j_{s_o d_o}^o+s_o)+\ldots+j_{kq}^o) = (b-\sum_{r=i_o}^{kq} j_r^o) - s_o < (i_o+s_o) - s_o = i_o$. So $\exists$ i_1, $o\leq i_1\leq (d-1)s$, $i_1+s > b-\sum_{r=i_1}^{kq} j_r^3$ (Choose $i_1 = i_o - s$, so $i_1\leq s_o d_o - s\leq (d-1)s$). From the definition of d_o and s_o, and from $(d-1)s < s_o d_o$, we now have that $(J_3,(s,d-1)) \notin S$. Contradiction.

This means that none of the packets that is stored at v_o can be accepted in w_o. Similarly none of the packets that is stored at w_o can be accepted in v_o. So G is deadlocked. This shows that our initial assumption that S is deadlock-free and $b<(k+1)q$ or not($S \subseteq \Phi SV(q,k,b)$) is false. □

For q=1 we have that for every deadlock-free, local, uniform (Φ,q,k,b) <u>count</u>-controller S (with q=1, $k\geq 1$, $b\geq 1$, Φ a strict and stable count-down function) $S\subseteq \Phi CV(1,k,b)$. (The proof is similar to the proof in [4], for the case that Φ=F.) For q>1 this result does not hold:

<u>Proposition 4.3.</u>

i) Let q>1, $k\geq 1$, $b\geq (q+1)k$, and let Φ be a count-down function. $S = \{ (m,(s,d)) \mid (((d+1)s\leq m) \text{ or } ((b-m)k\leq m-s)) \text{ and } (1\leq s\leq q) \text{ and } (o\leq d\leq k) \text{ and } (s\leq m\leq b)\}$ is a deadlock-free, local, uniform (Φ,q,k,b)-controller.

ii) If $b \leq (k+1)q+q-2$, then $\Phi CV(q,k,b) \subset S$

iii) If $b > (k+1)q+q-2$, then $\Phi CV(q,k,b) \equiv S$

Proof.

i) Let q,k,b,Φ be given. It is clear that $\Phi CV(q,k,b) \subseteq S$. If $S \subseteq \Phi SV(b,k,q)$, then we have by theorem 4.2. that S is deadlock-free.

Suppose that not($S \subseteq \Phi SV$). Then there are s,d and a statematrix J, ($1 \leq s \leq q$, $o \leq d \leq k$), with $(J,(s,d)) \in S$, and for $\vec{j} = \vec{j}(J)$, $\exists\, i_o$, $o \leq i_o \leq sd$, $i_o + s > b - \sum_{r=i_o}^{kq} j_r$ and $k . \sum_{j=o}^{kq} j_r < b - \sum_{r=o}^{kq} j_r - s$. Let s,d,J and i_o be given. We have that $(d+1)s \geq i_o + s > b - \sum_{r=i_o}^{kq} j_r \geq b - \sum_{r=o}^{kq} j_r \geq k . \sum_{r=o}^{kq} j_r + s$, hence $sd > k . \sum_{r=o}^{kq} j_r$. From $d \leq k$, it follows that $q \geq s > \sum_{r=o}^{kq} j_r$. Now we consider two cases.

CASE I : $i_o > k(q-1)$. From $\sum_{r=o}^{kq} j_r < q$, it follows that at a node with statematrix J, there are no packets stored with size q, hence for all $r \geq k(q-1)+1$ one has $j_r = o$. So $\sum_{r=i_o}^{kq} j_r = o$. Now $kq \geq sd \geq i_o \geq b-s \geq (k+1)q-s \geq kq$. Contradiction.

CASE II : $i_o \leq k(q-1)$. Now $kq \geq i_o + q \geq i_o + s > b - \sum_{r=i_o}^{kq} j_r \geq b - \sum_{r=o}^{kq} j_r > b-q \geq kq$. Contradiction.

We conclude that $S \subseteq \Phi SV(q,k,b)$, and hence that S is deadlock-free.

ii) Let $b \leq (k+1)q+q-2$. It is clear that $\Phi CV(q,k,b) \subseteq S$. We now show that not($\Phi CV(q,k,b) \supseteq S$). Controllers $\Phi CV(q,k,b)$ and S allow a node v with all buffers empty to accept a packet with size q-1. Now try to accept a packet with state (q,k). Notice that $(k+1)q > b-(q-1) = m$, hence $(m,(q,k)) \notin \Phi SV(q,k,b)$ and $(b-m)k = (q-1)k \leq b-3q+2 \leq m-q$, hence $(m,(q,k)) \in S$. S allows a node with $b-(q-1)$ free buffers to accept a packet with state (q,k), $\Phi CV(q,k,b)$ does not.

iii) Let $b > (k+1)q+q-2$. Again it is clear that $\Phi CV(q,k,b) \subseteq S$. We now show that $\Phi CV(q,k,b) \supseteq S$. Suppose $(m,(\Phi,s)) \notin \Phi CV(q,k,b)$ and $(m,(\Phi,s)) \in S$, for some m,Φ,s, $o \leq m \leq b$, $o \leq \Phi \leq k$, $1 \leq s \leq q$. Then $(k+1)s > m$

and $(b-m)k+s \leq m$. It follows that $(k+1)s > (b-m)k+s$, hence $s>b-m$ and $q>b-m$. Now $(k+1)q \geq (k+1)s > m > b-q$, hence $b \leq (k+1)q+q-2$, contradiction. □

It is presently open to find an "optimal" deadlock-free, local, uniform (Φ,q,k,b)-count-controller for $q>1$.

5. Final comments.

i) It is possible to vary the number of buffers at each node. The results of this paper can easily be generalized for the case that the condition $b \geq (k+1)q$ is replaced by the condition: for each node v, the number of buffers of v $b(v)$ is at least $\max\{(\Phi(p,v,t)+1)s(p) \mid p$ is a packet, and v is possibly on the path of $p\}$.

ii) The controllers ΦCV and ΦSV we presented in this paper have an important drawback: they do not prevent lifelock. Lifelock occurs when there are packets that will never reach their destination. An example of a network with lifelock is given in fig. 5.1. Assume $k \geq 2$, $q \geq 3$, $b \geq (k+1)q$ to be given, and suppose the controller FSV(b,k,q) is used.

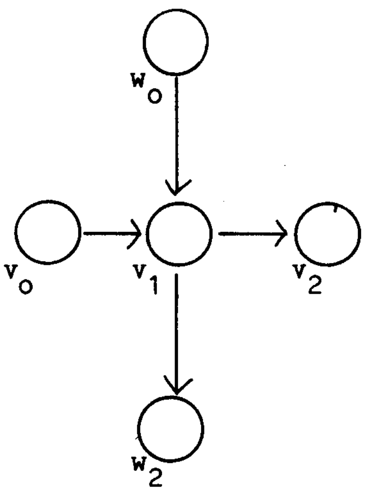

fig. 5.1.

In v_0 b-1 packets with destination v_2 and size 1 are created, and accepted in v_1. Now a packet p_1 with size 3 is created in w_0, with destination w_2. w_0 cannot be accepted in v_0. Now one packet in v_1 passes to v_2 and is consumed there; still w_0 cannot be accepted in v_0. Now a new packet with size 1 and destination v_2 is created in v_0 and is accepted in v_2. Again a packet in v_1 passes to v_2, etc. So there is an infinite sequence of moves, that prevents p_1 to move from w_0 to v_1. Hence packet p_1 is lifelocked.

Toueg [5] showed that FS(b,k) and BS(b,k) can be modified, such that the resulting controllers are lifelock free too. We will give a short informal description of the modification. The state of a packet now consists of the value F(p) (or B(p)) and the time that it was created, that is: the first moment that a processor tried to let the packet enter the buffers of a node v. (The size of the packet is 1.) A processor v accepts a packet p, if and only if the move is allowed according to controller FS(b,k) (or BS(b,k)), and there is no other packet p' waiting to be accepted by v, with an earlier creating time, for which the acceptance of p' by v is also allowed by FS(b,k) (or BS(b,k)). These controllers are called FSET(b,k) and BSET(b,k), respectively. The same modification can also be applied if we use other count-down functions than F or B. However, the same modifications to controllers ΦCV(q,k,b) or ΦSV(q,k,b) with q>1 do not yield lifelock-free controllers. A straightforward, but not very interesting way to find lifelock- and deadlock-free controllers that allow the packet size to vary, is to treat each packet as if its size is q and then use basically FSET($\lfloor \frac{b}{q} \rfloor$,k) or BSET($\lfloor \frac{b}{q} \rfloor$,k). (Packets are treated as if they all have the same size.) It remains an interesting and challenging open problem to find lifelock and deadlock-free controllers, that make an (essential) use of the differences in packet sizes.

References.

[1] Gelernter, D., A DAG-based algorithm for prevention of store-and-forward deadlock in packet networks, IEEE Trans. Comput. 10 (1981) 709-715.

[2] Merlin, P.M. and P.J. Schweitzer, Deadlock avoidance in store and forward networks - I : Store-and-forward deadlock, IEEE Trans. Comm. 28 (1980) 345-354.

[3] Shyamasunder, R.K., A simple lifelock-free algorithm for packet switching, Science of Comp. Programming 4 (1984) 249-256.

[4] Toueg, S., and J.D. Ullman, Deadlock-free packet switching networks, SIAM J. Comput. 10 (1981) 594-611.

[5] Toueg, S., Deadlock and lifelock-free packet switching networks, 12th Symp. on Theory of Computing, (STOC), 1980, 94-99.