A saturated tree network of polling stations with flow control

Paul Beekhuizen Jacques Resing

December 8, 2009

Abstract

We consider a saturated tree network with flow control. The network consists of two layers of polling stations, and all polling stations use the random polling service discipline. We obtain the equilibrium distribution of the network using a Markov chain approach. This equilibrium distribution can be used to efficiently compute the division of throughput over packets from different sources. Our study shows that this throughput division is determined by an interaction between the flow control limits, buffer sizes, and the service discipline parameters. A numerical study provides more insight in this interaction. The study is motivated by networks on chips where multiple masters share a single slave, operating under flow control.

1 Background

Primarily motivated by networks on chips, we study a saturated concentrating tree network of polling stations with flow control. Networks on chips are an emerging paradigm for the connection of on-chip modules such as processors and memories. Such modules are traditionally connected via single buses, but because these buses cannot be used by multiple modules simultaneously, communication difficulties arise as the number of modules increases. Networks on chips have been proposed as a solution (see [8]). In networks on chips, routers are used to transmit packets to their destination, so that multiple links can be used at the same time and communication becomes more efficient.

We are in particular motivated by a network on chip with multiple masters (e.g., processors) sharing a single slave (e.g., memory). Packets travel from the processors to the memory over a number of routers. Each router has several queues sharing a single link connecting that router to the next. Because the link can be seen as a server attending multiple queues, the router can be seen as a polling station, and the network of routers as a concentrating tree network of polling stations.

Open concentrating tree networks of polling stations (i.e., without limitations on the number of packets from the same source) were previously analysed in [4, 5]. Networks on chips, however, also implement flow control [14]. Networks with flow control operating under heavy loads resemble networks with a fixed number of packets (i.e., *closed* networks); flow control limits the number of packets from the same source to a *maximum*, and heavy loads imply that packets are quickly replaced by new packets once they leave the network, which is modelled by keeping the number of packets from the same source *fixed*.

Consequently, we analyse the throughput in a closed network of polling stations in this paper. It can relatively easily be argued that the *total* throughput in our closed network of polling stations is fixed. One of the functions of flow control, however, is to achieve fairness [11]. We therefore study the *division* of throughput over packets from different sources, which depends on flow control limits, buffer sizes, and service disciplines.

Pioneering work on closed queueing networks in the context of networks with flow control was done by Reiser [16]. He modelled a computer network operating under window flow control as a closed queueing network and devised an efficient heuristic to compute throughput figures for large networks. Many other studies where networks with flow control are modelled as closed queueing networks have since been performed, see, e.g., Baccelli and Bonald [3], and Garetto et al. [10].

Besides the studies in [4,5], open networks of polling stations have been analysed in [6,7,12,15]. Altman and Yechiali [1] study a *closed single-station* polling system. In this system, after a packet has been served, it moves from queue i to queue j with a probability that depends on both i and j, regardless of the history. The analysis of Altman and Yechiali was later extended to a polling system with a combination of a fixed population and external arrivals by Armony and Yechiali [2] and to a polling system with breakdowns by Dror and Yechiali [9].

This paper is organised as follows: We describe the network in more detail in Section 2. In Section 3 we describe how the network can be modelled as a Markov chain, and we derive its equilibrium distribution. This equilibrium distribution is used to obtain throughput results in Section 4. We perform a numerical analysis of the throughput in Section 5. We draw conclusions in Section 6.

2 Model

We consider a concentrating tree network consisting of two layers of polling stations with flow control, as displayed in Figure 1. Node 0 is the last node of the network, and has N queues. Some of those queues are connected to other nodes and some are not. If queue i of node 0 is connected to another node, then that node is called node i. Node i, $i > 0$, has N_i queues.

Packets in node i and packets in queue i of node 0 are called type i packets. Type i packets are further subdivided into type i,j packets, $j = 1, \ldots, N_i$, such that the sub-type of the packet denotes the source of the packets (see Fig. 1). If queue i of node 0 is not connected to another node (i.e., if node i does not exist), we define $N_i = 1$.

The network we consider is saturated, which means that as soon as a type i,j packet leaves the network, another type i,j packet immediately arrives. The number of type i,j packets in the network is thus fixed and we denote the number of type i,j packets by $L_{i,j}$. We furthermore define $L_i = (L_{i,1}, \ldots, L_{i,N_i})$.

We assume that the network operates in discrete time. All packets and time slots are of unit size and packet departures and arrivals occur at the end of time slots, with departures before arrivals. The queues of node 0 are finite, and the size of queue i of node 0 is denoted by B_i. If queue i of node 0 is full, packets from node i are blocked. When node 0 transmits a type i packet, a new packet from node i is allowed to enter node 0 at the same time. Queue j of node i is large enough to store all type i,j packets.

All nodes in the network, including node 0, use the 'random polling' service discipline. With this service discipline, every queue has a fixed probability of being served, i.e., every time slot the server of node i, $i = 0, \ldots, N$, serves queue j with probability $P_i(j)$, regardless of what happened

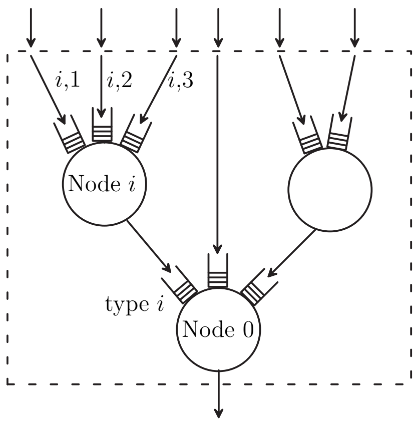


Figure 1: A tree network of polling stations with flow control. Due to saturation, the number of packets inside the dashed box is fixed.

in the past. The random polling service discipline was introduced by Kleinrock and Levy [13]. If some of the queues of node i are empty, we assume that the probability that queue j is served is proportional to $P_i(j)$, i.e., queue j is served with probability $P_i(j)/\sum_{l\in V_i} P_i(l)$, where V_i is the set of non-empty queues at node i. Within queues, packets are always served in FIFO order.

3 Markov chain analysis

In this section, we describe how node i and node 0 together can be modelled as a Markov chain and we obtain the equilibrium distribution of that Markov chain. This equilibrium distribution is used in Section 4 to determine the throughput of different sub-types of packets.

An important observation is that all nodes always try to transmit one packet per time slot if there is one. In particular, this implies that many packets may arrive to node 0 every time slot, but only one packet may leave. In stationarity, there will thus be as many type i packets in queue i of node 0 as possible, restricted by the buffer size B_i and the sum of the flow control limits $\sum_j L_{i,j}$.

In the sequel, we consider only those i for which node i exists; if node i does not exist, all type i packets are type i,1 packets. Moreover, in this case, type i,1 packets are served by node 0 with probability $P_0(i)$ because all queues of node 0 are always non-empty. The throughput of type i,1 packets is hence also equal to $P_0(i)$.

Suppose for a moment that queue i of node 0 is large enough to store at least all type i packets except one, i.e., $B_i \geq \sum_j L_{i,j} - 1$. Because all type i packets (except maybe one) are always in queue i of node, and this queue is served in FIFO order, all type i packets pass through the network in a cyclic manner; once an ordering of packet sub-types has been chosen, the sub-types will always be served in that order. The steady-state behaviour of the network thus depends on the initial configuration of packets and its equilibrium distribution is not unique. Moreover, in this case, the throughput can be calculated straightforwardly, as will be demonstrated in Section 4. In the sequel we therefore assume $B_i < \sum_j L_{i,j} - 1$.

If $B_i < \sum_j L_{i,j} - 1$, the state of queue i of node 0 only changes when the server is serving that queue; a new type i packet may only enter queue i of node 0 if another one leaves. For the computation of the throughput, we may thus disregard the times at which the server is at the other queues. We can determine the throughput in a network where node 0 is always serving queue i, and multiply that throughput by the fraction of time node 0 is actually serving queue i.

If node 0 is always serving queue i, node i and node 0 together form a closed tandem network with two nodes and unit service times, as displayed in Figure 2. The first node uses the random polling service discipline and has buffers large enough to store all packets, and the second node has a FIFO queue with a buffer of size B_i.

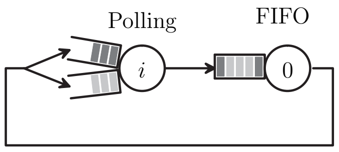


Figure 2: The closed tandem network.

We describe the state of this closed tandem network by a finite Markov chain. A state of this Markov chain is a length $B_i + 1$ vector $x = (x_1, \ldots, x_{B_i+1})$. Here, x_k, for $k = 1, \ldots, B_i$, describes the type of the packet at position k of the buffer of node 0, i.e., $x_1 = j$ if the first packet is a type i,j packet, and so on. The last element x_{B_i+1} describes the type of the packet that is going to be served by node i and will enter node 0 in the next time slot.

It is easily verified that this process is indeed Markovian: Node 0 always serves the first packet in the queue and the probability that node i serves queue j depends only on which queues are empty, which can be derived from x using the fact that there are $L_{i,j}$ type i,j packets in the network in total.

The state space of the Markov chain is given by:

$$\Omega_i = \left\{x \in \{1,\ldots,N_i\}^{B_i+1} : k_j(x) \le L_{i,j}\right\}, \tag{3.1}$$

where $k_j(x)$ is the number of j's in x. The condition $k_j(x) \le L_{i,j}$ reflects that there are at most $L_{i,j}$ packets in the buffer of node 0, and, moreover, if there are indeed $L_{i,j}$ packets in that buffer, the next packet served by node i (i.e., element x_{B_i+1}) must be of a different type.

We can now determine the equilibrium distribution of this Markov chain, denoted by $\pi_i(\cdot)$:

Lemma 3.1. *Suppose $B_i < \sum_j L_{i,j} - 1$. Recall that $k_l(x)$ is the number of l's in x. Then, for all $x \in \Omega_i$,*

$$\pi_i(x) = \frac{1}{C_i(B_i, L_i)} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x)}, \tag{3.2}$$

where $C_i(B_i, L_i)$ is the normalisation constant, given by

$$C_i(B_i, L_i) = \sum_{x\in\Omega_i} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x)}. \tag{3.3}$$

Proof. The equilibrium distribution is given by the unique normalised solution to the balance equations of the Markov chain:

$$\pi_i(x) = \sum_{j\in V_i(x)} \pi_i(j, x_1, \ldots, x_{B_i}) \frac{P_i(x_{B_i+1})}{\sum\limits_{k\in V_i(x)} P_i(k)}, \qquad x \in \Omega_i, \tag{3.4}$$

where $V_i(x)$ is the set of non-empty queues of node i in state x. The balance equations can be obtained by observing that every transition of the Markov chain consists of shifting packets at node 0 one buffer position ahead and by choosing the queue that will be served by node i in the next time slot. In other words, in state $(j, x_1, \ldots, x_{B_i})$, the vector $(x_1, \ldots, x_{B_i})$ is shifted towards positions 1 through B_i of the buffer. The probability that a type i, x_{B_i+1} packet is served at node i is given by the rightmost factor of (3.4).

Substituting Equation (3.2) in the right hand side of (3.4) yields

$$\begin{aligned}
\pi_i(x) &= \frac{1}{C_i(B_i, L_i)} \sum_{j\in V_i(x)} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x)} \frac{P_i(j)}{P_i(x_{B_i+1})} \frac{P_i(x_{B_i+1})}{\sum\limits_{k\in V_i(x)} P_i(k)} \\
&= \frac{1}{C_i(B_i, L_i)} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x)} \sum_{j\in V_i(x)} \frac{P_i(j)}{\sum\limits_{k\in V_i(x)} P_i(k)} \\
&= \frac{1}{C_i(B_i, L_i)} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x)},
\end{aligned}$$

which completes the proof. □

Example 3.2. Consider the situation with $B_i = 2$, and $L_i = (2, 2)$, and write $P_i(1) = p_1$ and $P_i(2) = p_2$. In this case, the state space Ω_i consists of the following six states: $(1, 1, 2)$, $(1, 2, 1)$, $(2, 1, 1)$, $(2, 2, 1)$, $(2, 1, 2)$, and $(1, 2, 2)$. In state $(1, 1, 2)$, two type i,1 packets occupy the two buffer positions and a type i,2 packet will move to the last buffer position in the next time slot. The first three states all have probability $p_1^2 p_2/C_i(B_i, L_i)$, and the last three all have probability $p_1 p_2^2/C_i(B_i, L_i)$. Because the sum of all probabilities is 1, it follows that $C_i(B_i, L_i) = 3p_1^2 p_2 + 3p_1 p_2^2 = 3p_1 p_2$.

Remark 3.3. The network studied in this section can also be seen as a variant of a classical urn problem: Consider N_i urns such that urn j contains $L_{i,j}$ balls, for $j = 1, \dots, N_i$. Suppose that $B_i + 1$ balls are drawn from different urns, and a ball from urn j is drawn with probability $P_i(j)/\sum_{l \in V_i} P_i(l)$ where V_i is the set of non-empty urns. After $B_i + 1$ balls have been drawn, the first ball is placed back in its original urn and a new ball is drawn. Then the second ball is placed back in its urn and a new ball is drawn, and so on. The equilibrium probability that, at an arbitrary point in time, the $B_i + 1$ most recently drawn balls originate from urns $x_1, \dots, x_{B_i+1}$ is given by $\pi_i(x)$.

The normalisation constant $C_i(B_i, L_i)$ can be computed efficiently using the following recursion:

$$C_i(b, (L_{i,1}, \dots, L_{i,n})) = \sum_{k=0}^{L_{i,n}} \binom{b+1}{k} (P_i(n))^k C_i(b-k, (L_{i,1}, \dots, L_{i,n-1})), \tag{3.5}$$

for any nonnegative integer $b \le B_i$, and any positive integer $n \le N_i$, with $\binom{b+1}{k} := 0$ for $k > b+1$.

This recursion can be obtained by conditioning on k, the number of type i,n packets in a system with buffer size b and flow control limits $(L_{i,1}, \dots, L_{i,n})$. In that system, there are $\binom{b+1}{k}$ states with k type i,n packets, each of which induces a factor $(P_i(n))^k$. Furthermore, given that there are k type i,n packets in a certain state x, the remaining $b+1-k$ vector positions of x are filled with remaining packet types, restricted by their flow control limits. This leads to a factor $C_i(b-k, (L_{i,1}, \dots, L_{i,n-1}))$, where

$$C_i(-1, (L_{i,1}, \dots, L_{i,n-1})) := 1,$$

i.e., if $k = b+1$, there are no remaining vector positions to be filled, and a factor 1 is required.

It might happen that $b - k \ge L_{i,1} + \ldots + L_{i,n-1}$. In this case, the $b+1-k$ remaining vector positions have to be filled with at most $L_{i,1} + \ldots + L_{i,n-1} < b+1-k$ packets. In other words, there are more positions in the vector than elements to fill the vector with, so we define:

$$C_i(b, (L_{i,1}, \dots, L_{i,n})) = 0, \qquad \text{for } b \ge L_{i,1} + \ldots + L_{i,n}, \text{ and all } n.$$

Finally, the behaviour on the boundary $n = 1$ follows from the definition of $C_i(\cdot,\cdot)$ (Eq. (3.3)):

$$C_i(b, (L_{i,1})) = (P_i(1))^{b+1}, \qquad \text{for } b < L_{i,1}.$$

4 Throughput computation

In this section, we determine the throughput of type i,j packets. As we argued in Section 3, this throughput is equal to the throughput in a network where node 0 always serves queue i, multiplied by the fraction of time node 0 is actually serving queue i. Because all queues of node 0 are always non-empty, node 0 serves queue i with probability $P_0(i)$, so we obtain:

$$\gamma_{i,j} = P_0(i) \widetilde{\gamma}_{i,j}, \tag{4.1}$$

where $\widetilde{\gamma}_{i,j}$ is the throughput of type i,j packets in a network where node 0 is always serving queue i.

It thus remains to determine $\widetilde{\gamma}_{i,j}$. First, we argued in Section 3 that if $B_i \ge \sum_j L_{i,j} - 1$, all packets move cyclically through the network without any change in their ordering. This implies that the fraction of time a type i,j packet is served is equal to the fraction of type i packets that are of type i,j, i.e., $\widetilde{\gamma}_{i,j} = L_{i,j}/\sum_k L_{i,k}$. Hence:

Proposition 4.1. *If* $B_i \ge \sum_j L_{i,j} - 1$,

$$\gamma_{i,j} = P_0(i) \frac{L_{i,j}}{\sum\limits_k L_{i,k}}. \tag{4.2}$$

If $B_i < \sum_j L_{i,j} - 1$, the throughput of type i,j packets can be determined using the results of Section 3:

Proposition 4.2. *If $B_i < \sum_j L_{i,j} - 1$,*

$$\gamma_{i,j} = P_0(i)P_i(j)\frac{C_i(B_i - 1, L_i - e_j)}{C_i(B_i, L_i)}, \tag{4.3}$$

where e_j is a vector of which all elements are zero, except the jth, which is 1.

Proof. The throughput of type i,j packets is given by the probability that a type i,j packet is served by node 0:

$$\widetilde{\gamma}_{i,j} = \sum_{\substack{x\in\Omega_i \\ x_1=j}} \pi_i(x) = \frac{1}{C_i(B_i, L_i)} \sum_{\substack{x\in\Omega_i \\ x_1=j}} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x)}. \tag{4.4}$$

For all x over which the sum is taken, the first element is a j. The remaining elements are restricted by the flow control limits, but are otherwise chosen arbitrarily. We thus obtain:

$$\begin{aligned}\widetilde{\gamma}_{i,j} &= \frac{1}{C_i(B_i, L_i)} P_i(j) \sum_{\substack{x\in\Omega_i \\ x_1=j}} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x_2,\ldots,x_{B_i+1})} \\ &= \frac{C_i(B_i - 1, L_i - e_j)}{C_i(B_i, L_i)} P_i(j).\end{aligned}$$

Equation (4.3) follows by combining the above expression with (4.1). □

In case the buffer sizes are small enough, we can reduce Equation (4.3) to a much simpler expression:

Proposition 4.3. *If $B_i < L_{i,j}$ for all j,*

$$\gamma_{i,j} = P_0(i)P_i(j). \tag{4.5}$$

Proof. If $B_i < L_{i,j}$ for all j, the number of j's in $(x_1, \ldots, x_{B_i+1})$ can never exceed $L_{i,j}$. The restriction that $k_j(x) \leq L_{i,j}$ can thus be removed from the definition of Ω_i, the state space of the Markov chain. It then follows from Newton's binomium that $C_i(B_i, L_i) = (\sum_j P_i(j))^{B_i+1} = 1$, and likewise $C_i(B_i - 1, L_i - e_i) = 1$. Equation (4.3) then yields $\gamma_{i,j} = P_0(i)P_i(j)$.

A more intuitive explanation is the following: The throughput of type i,j packets, $\widetilde{\gamma}_{i,j}$, is also given by the probability that node i serves queue j. If $B_i < L_{i,j}$ for all j, queue i of node 0 is too small to store all type i,j packets, for any j, so all queues of node i are always non-empty. Queue j is thus served by node i with probability $P_i(j)$. □

Using these throughput results, the mean total sojourn time and the number of packets in queue i of node 0 can be found easily using Little's law. As there are $L_{i,j}$ type i,j packets in the network, the total mean sojourn time is equal to

$$\mathbb{E}[T_{i,j}] = \frac{L_{i,j}}{\gamma_{i,j}}.$$

Furthermore, every type i packet spends on average $B_i/P_0(i)$ time slots in queue i of node 0, so the mean number of type i,j packets in that queue is given by

$$\mathbb{E}[Q_{i,j}^{(0)}] = \gamma_{i,j}\frac{B_i}{P_0(i)} = B_i\widetilde{\gamma}_{i,j}. \tag{4.6}$$

5 Numerical analysis

The analysis of the previous sections illustrates that the division of throughput is determined by an interaction between the flow control mechanism and the service disciplines: For small buffers, the throughput division is determined by the service disciplines, for large buffers by the flow control mechanism, and for intermediate buffers by a mix of the two. For intermediate buffers the throughput division can be determined using Equation (4.3). This equation by itself, however, offers little insight in the precise interaction between the flow control limitations and the service disciplines. In this section, we therefore study how the throughput depends on the different parameters through a numerical study.

We consider a network in which node i has four queues. We focus on the throughput given that node 0 is always serving queue i, i.e., on $\widetilde{\gamma}_{i,j}$. The actual throughput in the network can easily be found by multiplying that throughput by the probability that node 0 is serving queue i. The running example of this section has the following parameters: $B_i = 32$, $P_i = (0.1, 0.2, 0.3, 0.4)$, and $L_i = (20, 16, 12, 8)$. We vary one of these parameters at a time, and study how this affects the throughput division.

First, we show the throughput division in Figure 3 for B_i ranging from 1 to 64. This figure clearly illustrates that the throughput is determined by the service disciplines for small buffers, and by the flow control mechanism for large buffers. However, the transition from the small buffer regime to the large buffer regime does not occur monotonously; the throughput of some packet types first increases and later decreases.

We can explain this phenomenon by looking at the number of packets in queue i of node 0. As the buffer size grows, the number of type i,j packets in queue i of node 0 gets closer to the maximal number allowed by flow control. The flow control restrictions thus cause some queues to remain empty and the throughput of corresponding packet types to decrease. If some queues are empty, the other non-empty queues are served with a higher probability and the throughput of the corresponding packet types will increase. As the buffer size grows even further, the flow control restrictions on these packet types will become stronger as well, which can cause their throughput to fall again.

For example, if B_i is equal to twelve, the mean number of type i,4 packets in queue i of node 0 is roughly five (see Equation (4.6)), out of eight allowed by flow control. On the other hand, the mean number of type i,3 packets in queue i of node 0 is roughly four, out of twelve allowed. The flow control restrictions on type i,4 packets are thus more severe than those on type i,3 packets. As the buffer size B_i increases further, the throughput of type i,4 will hence decrease and that of type i,3 will grow.

When B_i is twenty-five, there are on average roughly eight type i,3 packets in queue i of node 0, out of twelve allowed. Flow control restrictions have thus become more severe for this type, and its throughput will decrease again.

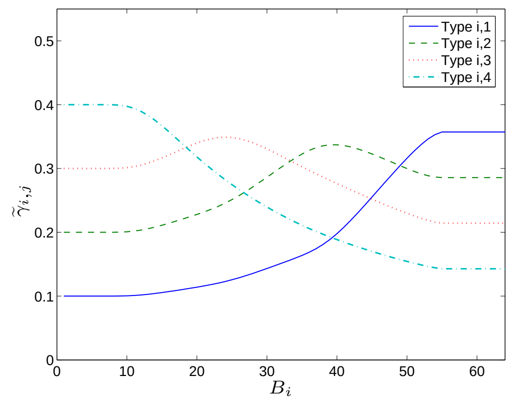


Figure 3: The division of throughput for various buffer sizes.

	$j=1$	$j=2$	$j=3$	$j=4$
$P_i(j)$	0.1	0.2	0.3	0.4
$L_{i,j}$	20	16	12	8
$\widetilde{\gamma}_{i,j}$	0.1512	0.3016	0.3198	0.2274
$\mathbb{E}[Q_{i,j}^{(0)}]$	4.84	9.65	10.23	7.28
$\mathbb{E}[Q_{i,j}^{(0)}]/L_{i,j}$	0.24	0.60	0.85	0.91

Table 1: Parameters and performance measures of the running example.

That some packet types benefit from the flow control restrictions on other packet types is especially clear for type i,1 packets. Queue 1 of node i is served with a 0.1 probability, and only when the other queues are frequently empty due to the flow control restrictions, type i,1 packets will receive a more significant part of the throughput. Figure 3 clearly illustrates this: The throughput of type i,1 experiences a sharp increase when the throughputs of all other types decrease.

We consider again the running example, i.e., we set $B_i = 32$. In Table 1, we show $\mathbb{E}[Q_{i,j}^{(0)}]$, the expected number of type i,j packets in node 0, as well as $\mathbb{E}[Q_{i,j}^{(0)}]/L_{i,j}$, the expected number of type i,j packets in node 0 relative to its maximum. Based on the discussion above, the latter can be seen as a (rough) measure for how strong the flow control restrictions are; the closer to 1, the stronger the flow control restrictions. Table 1 reveals that type i,1 packets are not strongly restricted by flow control, but type i,4 packets are; the expected number of packets in node 0 relative to its maximum is 0.24 and 0.91 for type i,1 and type i,4 respectively.

We proceed by investigating the sensitivity of the throughput of these packet types on changes in the flow control limits and service disciplines for the running example, i.e., we vary one of the values of $L_{i,j}$ and $P_i(j)$ for $j = 1, 4$. Here, the value of $P_i(j)$ is varied for one specific j, while maintaining the *ratios* between the other values of $P_i(k)$, for $k \neq j$, and maintaining a total sum of 1.

Figures 4a and 4b show the throughput of type i,1 packets as a function of $P_i(1)$ and $L_{i,1}$, respectively. For both figures, a vertical line marks the value on the x-axis corresponding to the running example. Clearly, the throughput of type i,1 packets is not sensitive to a change in its flow control limit but it is sensitive to a change in the service discipline; the throughput hardly changes if the flow control limit is changed but it does change if the service discipline is changed.

In contrast, Figures 5a and 5b show the sensitivity of type i,4 throughput on changes in its flow control limit and the service discipline, where the running example is again marked by a vertical line. The throughput of type i,4 is not sensitive to a change in the service discipline, but it is sensitive to a change in the flow control limit.

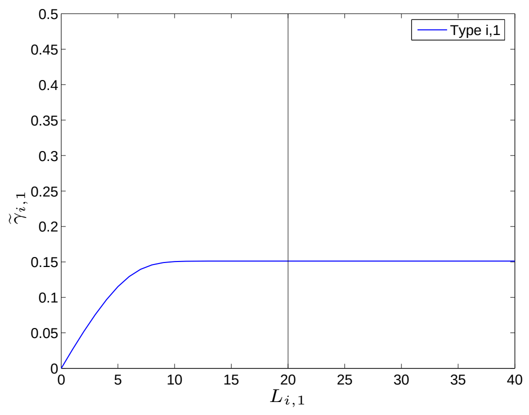
These observations can be used to decide whether flow control limits or service disciplines have to be adjusted when the throughput division over various packet types has to be changed. If a packet type is strongly restricted by the flow control mechanism, the flow control limits should be changed, and otherwise the service discipline of node i should be changed.

6 Conclusion

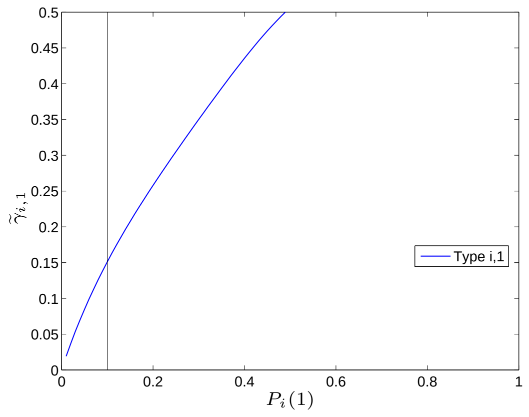
We modelled a saturated concentrating tree network with flow control as a closed network. The equilibrium distribution of the Markov chain describing this network can be derived in closed form for the random polling discipline. Using the results from the Markov chain approach, the division of throughput over various packet types can be determined.

For small buffers, the throughput division over packet types is fully determined by the service disciplines, and for large buffers by the flow control mechanism. For intermediate buffers the throughput division is steered by an interaction between the two. We performed a numerical study that provides more insight in the specifics of this interaction.

The numerical study showed that the transition from the small buffer regime to the large buffer regime does not necessarily occur monotonously. If there are many packets of the same type in

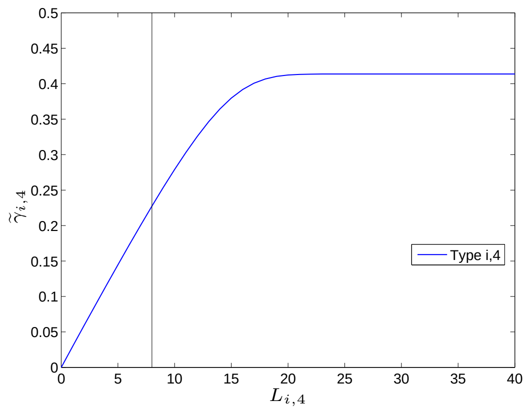


(a) Throughput as a function of $L_{i,1}$.

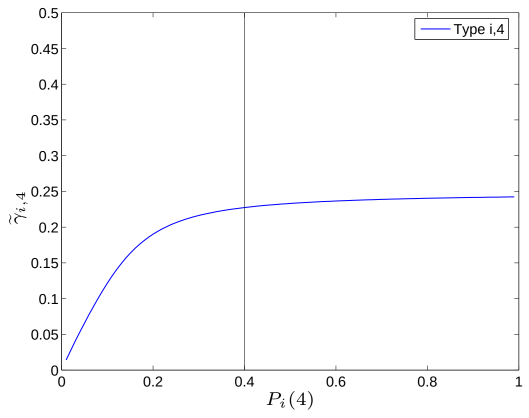


(b) Throughput as a function of $P_i(1)$.

Figure 4: The type i,1 throughput as a function of $L_{i,1}$ and $P_i(1)$. The vertical line corresponds to the 'running example'.



(a) Throughput as a function of $L_{i,4}$.



(b) Throughput as a function of $P_i(4)$.

Figure 5: The type i,4 throughput as a function of $L_{i,4}$ and $P_i(4)$. The vertical line corresponds to the 'running example'.

node 0, flow control restrictions become stronger, which limits the throughput of that type. The throughput that becomes available as a result is divided over the other packet types for which the flow control restrictions are less severe, which causes their throughput to rise. If the buffer size is increased even further, the number of packets of those types in node 0 also increases, which causes flow control limitations to become stronger and their throughput to decrease again.

The expected number of packets of different types in node 0 provides insight in the extent to which packet types are restricted by flow control, which in turn determines which parameters should be changed if the throughput division has to be adjusted: If almost all packets of the same type are in node 0, flow control plays a key role. Changing flow control limits will thus affect the throughput of that type, but changing the service discipline parameters will not. On the other hand, if not many packets are in node 0, service disciplines play a key role. Changing flow control limits thus has little effect, but changing service discipline parameters has a large effect.

In this paper, we only analysed the network with two layers of polling stations with random polling. The phenomena described above, however, occur in more complex tree networks as well; additional simulations have revealed that they also occur in networks of three layers and networks with the cyclic 1-limited service discipline.

References

[1] E. Altman and U. Yechiali, *Polling in a closed network*, Probability in the Engineering and Informational Sciences **8** (1994), pp. 327–343.

[2] R. Armony and U. Yechiali, *Polling systems with permanent and transient jobs*, Stochastic Models **15** (1999), no. 3, pp. 395–427.

[3] F. Baccelli and T. Bonald, *Window flow control in FIFO networks with cross traffic*, Queueing Systems **32** (1999), no. 1, pp. 195–231.

[4] P. Beekhuizen, T.J.J. Denteneer, and J.A.C. Resing, *End-to-end delays in polling tree networks*, Proc. of ValueTools, 2008.

[5] P. Beekhuizen, T.J.J. Denteneer, and J.A.C. Resing, *Reduction of a polling network to a single node*, Queueing Systems **58** (2008), no. 4, pp. 303–319.

[6] D. Bertsimas and J. Niño-Mora, *Optimization of multiclass queueing networks with changeover times via the achievable region approach: Part II, the multi-station case*, Mathematics of Operations Research **24** (1999), pp. 331–361.

[7] J.G. Dai and O.B. Jennings, *Stabilizing queueing networks with setups*, Mathematics of Operations Research **29** (2004), pp. 891–922.

[8] W.J. Dally and B. Towles, *Route packets, not wires: On-chip interconnection networks*, Proc. of the Design Automation Conference, 2001, pp. 684–689.

[9] H. Dror and U. Yechiali, *Closed polling models with failing nodes*, Queueing Systems **35** (2000), no. 1, pp. 55–81.

[10] M. Garetto, R.L. Cigno, M. Meo, and M. Ajmone Marsan, *A detailed and accurate closed queueing network model of many interacting TCP flows*, Proc. of IEEE INFOCOM, 2001, pp. 1706–1715.

[11] M. Gerla and L. Kleinrock, *Flow control: A comparative study*, IEEE Transactions on Communications **28** (1980), no. 4, pp. 553–574.

[12] O.B. Jennings, *Heavy-traffic limits of queueing networks with polling stations*, Mathematics of Operations Research **33** (2008), no. 1, pp. 12–35.

[13] L. Kleinrock and H. Levy, *The analysis of random polling systems*, Operations Research **36** (1988), no. 5, pp. 716–732.

[14] A. Rădulescu, J. Dielissen, K. Goossens, E. Rijpkema, and P. Wielage, *An efficient on-chip network interface offering guaranteed services, shared memory abstraction, and flexible network configuration*, Proc. of DATE, 2004, pp. 878–883.

[15] M.I. Reiman and L.M. Wein, *Heavy traffic analysis of polling systems in tandem*, Operations Research **47** (1999), no. 4, pp. 524–534.

[16] M. Reiser, *A queueing network analysis of computer communication networks with window flow control*, IEEE Transactions on Communications **27** (1979), no. 8, pp. 1199–1209.