The Capacity Allocation Paradox

Asaf Baron, Ran Ginosar and Isaac Keslassy
Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa, 32000, Israel
{ab@tx, ran@ee, isaac@ee}.technion.ac.il

***Abstract*— The Capacity Allocation Paradox (CAP) destabilizes a stable small-buffer network when a link capacity is increased. CAP is demonstrated in a basic 2x1 network topology. We show that it applies to fluid, wormhole and packet-switched networks, and prove that it applies to various scheduling algorithms such as fixed-priority, round-robin and exhaustive round-robin. Their capacity regions are modeled and surprising phenomena are described. For instance, once increasing a link capacity destabilizes a stable network, increasing it further to infinity might never restore stability. Further, we exhibit networks with arbitrarily tight link-capacity stability regions, in which any small deviation from an optimal link capacity might make the network unstable. Finally, we suggest ways to mitigate CAP, e.g. by using GPS scheduling.**

I. Introduction

NETWORK designers typically assume that adding capacity can only improve performance. Thus, the principal goal of network design is assumed to be finding the minimum capacity needed for acceptable performance. Beyond that minimum capacity, any capacity should work.

Such intuitive view is central to classical networking theory. For instance, queuing theory teaches us that increasing the service rate μ stabilizes most queues of arrival rate λ, as long as $\lambda < \mu$ [1]. Likewise, information theory shows that increasing the channel capacity C can help transmit most codes of information rate R with arbitrarily small block error, as long as $R < C$ [2]. More generally, the network *capacity region* is considered in such diverse networking fields as multi-hop wireless networks [3], queuing networks [4], packet switches [5], mobile ad-hoc networks [6], interconnection networks [7], networks-on-chip [8], and satellite transmissions [9].

There are two well-known theoretical exceptions to this rule, but network designers can easily avoid them. First, Braess's paradox states that increasing the capacity of a link may harm the performance of a network in which each source selfishly chooses its route [10]-[13]. But by making routing deterministic, Braess's paradox can be avoided altogether. Second, in networks with reentrant lines or similar cyclic dependency, given specific initial conditions, arbitration schemes and topologies, increasing the service rate of a queue might make the network unstable, as shown in [14] (see also [15]-[18]). However, most practical networks do not contain reentrant lines and do not satisfy the additional specific conditions. Thus, in practice, network designers seldom need to take these two exceptions into account.

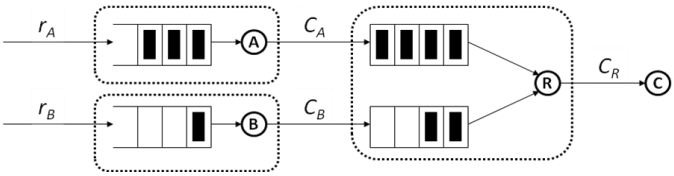

Figure 1: Simple 2×1 network

As a consequence, network capacity allocation algorithms are typically simple to design: given a target performance guarantee, network designers need only look at the flows that do not satisfy this performance target, and allocate more capacity on their paths. For instance, such an approach is employed to provide performances guarantees in hard-to-model wormhole networks such as spacecraft networks [19] and networks-on-chip [20]. It is assumed that offering increasingly more capacity would *necessarily* reach the target performance.

In this paper we revisit this classical assumption of networking theory, and show that it may be wrong. When some of the buffers in the network are *small*, we prove that adding capacity can make the network unstable, even when adding infinite capacity. Therefore, an uncontrolled capacity allocation algorithm may not converge.

This phenomenon is termed the *Capacity Allocation Paradox (CAP)*, and we prove that it is quite general in finite-buffer networks. For instance, Figure 1 illustrates a simple 2×1 network with two sources A and B, a router R, and a destination C. The two sources A and B generate packets at respective rates r_A and r_B. They send the packets to their corresponding small finite buffers at the router R using links of capacities C_A and C_B. In turn, R schedules the packets and forwards them to the destination C on a link of capacity C_R. If the finite router buffers are full, the corresponding sources queue their packets at their infinite queues until the buffers are not full anymore. For instance, in Figure 1, A cannot send its packets because its buffer is full, while B can send its queued packet. The network is considered stable if the expected queue sizes are bounded.

We demonstrate that *increasing C_A or C_B can destabilize a stable network*. For instance, we might want to improve the average delay of flow A and change the capacity allocation, by

increasing C_A while keeping other parameters constant. We show that this can make the network unstable by overflowing the queue of B. More dramatically, even if we keep increasing $C_A \to \infty$, the network remains unstable.

CAP is a critical issue in finite-buffer networks, because it applies in general settings. For instance, it can happen using any number of sources, since it already happens with two sources. More generally, it applies to any finite-buffer network topology in which this 2×1 example can be embedded. It can also occur with any finite buffer size, though of course it is more acute for smaller buffers. Last, we prove that it applies to many practical router scheduling schemes, such as fixed-priority, round-robin and exhaustive round-robin.

Buffer size is limited in many networks, such as networks-on-chip [20], which need to employ small buffers because of their area and power requirements, as well as computer interconnection networks [7] and spacecraft networks [19][21]. The prevalence of such small-buffer networks emphasizes the importance of considering CAP issues.

Three different settings are used to demonstrate the generality of CAP. First, we analyze a bufferless router with *fluid traffic*. That case provides some intuition into the underlying reasons behind CAP, as well as into the mathematical tools used to analyze it. *Wormhole-switched* networks, in which packets are broken into small flits and routers can switch a flit without waiting for the others to arrive, and which are quite useful in finite-buffer networks, are discussed next. The CAP phenomenon is demonstrated in wormhole-switched networks, and their stability region under several work-conserving scheduling schemes is characterized. We show how in some stable networks, increasing C_A to infinity while keeping other parameters constant makes the network unstable and cannot restore stability. We also prove that the size of the stability region for C_A can be arbitrarily small. In other words, any small deviation from an optimal C_A, either upwards or downwards, will make the network unstable. Last, we propose mitigations of the CAP phenomenon and prove that a GPS scheduling algorithm is guaranteed to provide stability in the admissible capacity region (i.e., all capacities above packet rates)

Finally, we consider a third setting with classical *packet-switched* store-and-forward networks. We model these as *dropping* instead of blocking networks, and assume that dropped packets need to be resent. We demonstrate the existence of the CAP phenomenon in such a network as well.

Once a capacity increase makes a network unstable, we further illustrate a few ways to bring it back to stability. An unintuitive solution is to reduce the link capacities of the stable flows. Likewise, it is possible to add capacity and buffering on the links of the unstable flows. Last, another solution is to change the arbitration policy, e.g. by adopting a GPS arbitration.

The rest of this paper is organized as follows. First, in Section II, we give some intuition for the CAP phenomenon based on the fluid traffic setting. Then, in Section III, we analyze wormhole-switched networks and show the existence

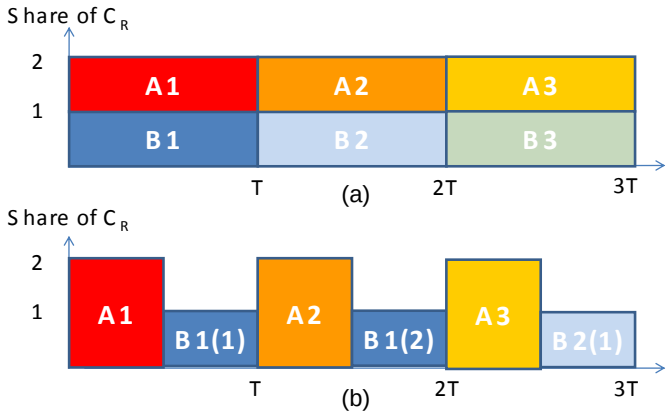

Figure 2: Link R sharing. C_B=1, C_R=2, packet arrival rate at A,B is 1 [pkt/T]. (a) C_A =1, entire traffic transferred on link R. (b) C_A =2, A interrupts B.

of the CAP as well. We model packet-switched networks in Section IV and prove that CAP occurs in these. Last, Section V presents simulation results and discussions of our models. For space reasons, we refer interested readers to the technical report [22] for the full proofs of all results in this paper.

II. Intuition

A bufferless example is used to provide intuition into the CAP phenomenon. Consider the example of Figure 1 with bit-by-bit switching and *no buffer space* in the router, i.e. just enough space to switch one bit. Assume that time is slotted, and that both nodes A and B receive a packet of size 1 at the start of each time-slot. Further assume that traffic from node A is of higher priority, and neglect propagation times.

We show that both the nodes A and B manage to send their packets given some initial capacity allocation. Subsequently, as we increase one of the link capacities, they cannot both do it anymore and the network becomes unstable.

A. *Low Capacity – Stable Network*

Figure 2(a) illustrates how nodes A and B share the capacity of link R, C_R, when C_A=C_B=1 and C_R=2. For this capacity allocation C_R=C_A+C_B and therefore any traffic arriving at the router can be immediately transferred and there is no blocking. Each node A,B can produce traffic at full rate of 1 and send the packet that arrived at the beginning of each time-slot by the end of the same time-slot.

B. *High Capacity – Unstable Network*

Consider C_A=2. Now C_A=C_R, and therefore once node A transmits its packet it uses the full router output link bandwidth. Since A has higher priority, its traffic is not interrupted and thus node B cannot send anything. At each time-slot n, A sends its packet at rate 2, completing by time n+1/2.

Figure 2(b) illustrates how nodes A and B share the capacity C_R for this latter capacity allocation. The rate of node B is bounded by C_B=1 and therefore B is unable to complete sending its packet. B only sends half a packet by the end of time-slot n, and the other half of the packet remains in the queue in node B. For example, during [0,T] node B sends only

B1(1), the first half of packet B1, and in the next time slot it sends the other half, *B1(2)*. Thus, the queue in B grows by half a packet each time slot. As a result, due to this increased link capacity, the network is now unstable, illustrating the CAP phenomenon.

C. Fluid Model

A deterministic continuous model of the system can be described as follows. For $i \in \{A,B\}$, let (Q_i,A_i,D_i) denote the cumulative amount of data queued, arrived, and departed respectively for node i by time t. Then we have,

$$Q_i(t) = Q_i(0) + A_i(t) - D_i(t) \quad (1)$$

$$A_i(t) = \lceil t \rceil \quad (2)$$

$$\dot{D}_A(t) = \min\{C_A, C_R\} \text{ if } Q_A(t) > 0 \quad (3)$$

$$\dot{D}_B(t) = \min\{C_B, C_R - \dot{D}_A(t)\} \text{ if } Q_B(t) > 0 \quad (4)$$

Equation (1) expresses the flow conservation at node i. Equation (2) reflects the periodic arrivals of packets of size 1. Equation (3) states that packets from A can leave at a rate limited by the minimum of the link capacity of A and the available router output link capacity. Finally, Equation (4) limits the rate at which packets leave B to the minimum of the link capacity of B and the available router output link capacity, given that A has higher priority.

Let's focus on the case of $C_A \le C_R$. When $Q_B(t) > 0$, substituting (3) into (4) yields:

$$\dot{D}_B(t) = \begin{cases} \min\{C_B, C_R - C_A\} & Q_A(t) > 0 \\ C_B & else \end{cases} \quad (5)$$

Observe that B is constrained by two limits: its link capacity and the residual bandwidth of the router link after A has been served. As C_A increases, node A exploits a higher portion of the router output bandwidth, but for a smaller portion of the time. Equation (5) shows these two contradicting influences of C_A on the service rate of node B. Assuming A is stable, $Q_A(t)>0$ for $1/C_A$ of the time. Thus, the average departure rate of node B is given by:

$$\begin{aligned} E\left[\dot{D}_B(t)\right] &= \frac{1}{C_A}\min\{C_B, C_R - C_A\} + \left(1 - \frac{1}{C_A}\right)C_B \\ &= C_B - \max\left(\frac{C_A + C_B - C_R}{C_A}, 0\right) \end{aligned} \quad (6)$$

In our example, for stability we need $E\left[\dot{D}_B(t)\right] \ge 1$. Since C_B=1, C_R=2 and $E\left[\dot{D}_B(t)\right]$ = 1-max(1-1/C_A,0) = min(1,1/C_A), the network is unstable whenever C_A>1. Therefore, any increase of C_A beyond 1 activates the CAP phenomenon and the network becomes unstable.

III. CAP in Wormhole Networks

A. Model and Notations

In *wormhole-switched* networks, packets are broken into small flits and routers can switch a flit without waiting for the others to arrive. This is in contrast to networks that use store and forward where the router should wait for the entire packet to arrive before transmitting it. Thus, wormhole switching enables the use of small buffers and achieves smaller delay at low load. If a given node cannot forward its flits, for example because the buffer in the next node on the path is full, then it blocks all the flits trailing it, and thus a packet can spread along the network (hence the name wormhole).

Consider the network illustrated in Figure 1. Assume that the packet creation process in node i is stationary and ergodic of average rate r_i. Let L denote the constant number of flits per packet, and R_i denote the average rate of flits created in node i. Then $R_i = r_i \cdot L$. For $i \in \{A,B,R\}$, let link i denote the link going out of node i (for example, link R is the link from the router to node C). As in Figure 1, let C_A, C_B, and C_R respectively denote the flit transmission capacities of links A, B and R. Finally, for $i \in \{A,B\}$, let *queue i* denote the queue in node i, which we assume infinite, and let *buffer i* denote the router buffer storing packets from node i. Buffer i is assumed finite, of size B_i>0.

We assume that the buffer size is much smaller than the number of flits per packet, namely $B_A, B_B << L$ (typically, a wormhole buffer may include 4-16 flits, while packets may comprise up to 1000 flits). Note that one of the main advantages of wormhole networks is that they require small buffers in the routers, as reflected by this assumption. We also assume that there are at least two virtual channels from the router to node C, and therefore do not consider deadlocks [7]. Considering flit rates rather than bit rates in the following accounts for the virtual channel overhead. Finally, we neglect all propagation delays, and assume that a node knows when there is buffer space available at the next node, thus eliminating the use of credits in this analysis.

Define the maximum utilization U_i of link $i \in \{A,B\}$ as the utilization achieved on link i when queue i is never empty. Then queue i is defined to be *stable* iff $U_i \cdot C_i > R_i$ [23].

We consider three common work conserving arbitrations as follows:

1) *Exhaustive Packet Round Robin (EPRR)* [24]: The router transmits a packet from one input port until the buffer becomes empty or until the last flit of a packet is sent. Then the router moves to serve the other input port.
2) *Round Robin Per Flit (RRPF)* [25][26]: The router transmits one flit from one of the buffers and then serves the other buffer, interleaving flits from the two buffers.
3) *General Processor Sharing (GPS)* [27], i.e. Packetized GPS (PGPS) applied to flits: The router output bandwidth is divided between the two flows according to their respective flit rates, namely $C_R R_A/(R_A+R_B)$ and $C_R R_B/(R_A+R_B)$. Priority is then assigned dynamically according to the flit completion time in bit-by-bit GPS.

The CAP phenomenon is shown to exist for EPRR and RRPF arbitrations, whereas it does not occur when using GPS. In addition, it can be seen that for some capacity allocation EPRR behaves as if the router grants priority to packets from one of the nodes. Therefore, the discussion of EPRR stability (Section C below) provides some intuition about the CAP phenomenon for priority-based arbitrations as well.

B. Necessary Stability Conditions

To maintain stability, a link should be capable of

transmitting all traffic presented to it. The following conditions are necessary for stability:

$$C_A > R_A \quad (7)$$
$$C_B > R_B \quad (8)$$
$$C_R > R_A + R_B \quad (9)$$

As shown below, in some cases these necessary conditions are insufficient.

C. Stability of EPRR

Assume that the router uses EPRR arbitration and that the *necessary stability conditions are fulfilled*. We break the discussion of the stability conditions into several cases according to the relations among link capacities, and show that for some capacity allocations the network is stable, while increasing one of the links capacities may destabilize the network.

Case (1): $C_A + C_B \le C_R$. In this case, the router output link transmits flits at a higher rate than they arrive at the router, even if both links A and B are active.

Theorem 1: When $C_A + C_B \le C_R$, the necessary stability conditions are also sufficient.

Proof: Since the router output link capacity is no less than the sum of input link capacities, the buffers will hold no more than one flit, and the queue of node *i* gets serviced at full link capacity. If the necessary stability conditions are fulfilled, the service rate of both queues exceeds their average flit generation rate, and thus both queues are stable, as well as the entire network. ■

Case (2): $C_A \ge C_R$ *and* $C_B < C_R$. *In this case, the router output link transmits flits slower than link A but faster than link* B. We show that while queue A is stable, the necessary stability conditions may not be sufficient for queue B.

Theorem 2: The necessary stability conditions are also sufficient for the stability of queue A.

Proof: Link R is busy transmitting packets from node B for a portion of the time of at most $\beta=R_B/C_R$. In the worst case during these busy periods the entire path from node A to node C is idle, and during the remainder of the time the router is able to transmit flits from node A. Therefore, a sufficient condition for the stability of queue A is:

$$(1-\beta)\cdot\min\{C_A,C_R\} > R_A \underset{C_A\ge C_R}{\Leftrightarrow} \left(1-\frac{R_B}{C_R}\right)\cdot C_R > R_A$$
$$\Leftrightarrow \underbrace{C_R > R_A + R_B}_{\text{Necessary Condition}}$$

■

Consider the conditions for stability of queue B. We show that the transmission of a packet from A is not interrupted, and that there are cases in which the maximum utilization of link B is less than 100%.

Theorem 3: Once the transmission of a packet from node A on link R has started, it is not interrupted until the entire packet has been transmitted.

Proof: Since $C_A \ge C_R$, buffer A does not become empty as long as the packet transmission has not ended. Thus, using the EPRR arbitration, the packet transmission is not interrupted. ■

Theorem 4: Assume that the necessary stability conditions hold, and denote $x^+=\max(x,0)$. Then the maximum utilization of link B is

$$U_B = 1 - r_A\left(\frac{L}{C_R} - \frac{B_B}{C_B}\right)^+ \quad (10)$$

Proof: By definition, U_B is computed when queue B is always non-empty. *Theorem 3* shows that every time a packet from A is transmitted on link R, link B can transfer data only until it fills its buffer in the router. Afterwards, and until the transmission of the packet from A has ended, link B is stalled. The time it takes for buffer B to fill up is B_B/C_B, and the time it takes to send a packet from A is L/C_R. Therefore, during the transmission of a packet from A, node B is stalled for $\left(L/C_R - B_B/C_B\right)^+$. Since the packet rate from node A is r_A, the portion of time that link B is stalled is $r_A\left(L/C_R - B_B/C_B\right)^+$, yielding Equation (10). ■

The following theorems present the stability conditions for queue B.

Theorem 5: If the necessary stability conditions hold, queue B is stable iff

$$C_B > \frac{R_B}{1 - r_A\left(\frac{L}{C_R} - \frac{B_B}{C_B}\right)^+} \quad (11)$$

Proof: By definition, queue B is stable iff $U_B \cdot C_B > R_B$. Using the result of *Theorem 4*, this condition can be rewritten as $C_B \cdot \left(1-\left(L/C_R - B_B/C_B\right)^+\right) > R_B$, hence the result. ■

For the following, define $\pi=B_B/L$ (the portion of a packet that fits in buffer B) and $\gamma=R_B-B_B\cdot r_A=R_B-\pi\cdot R_A$.

Theorem 6: In Case (2), if the necessary stability conditions hold then queue B is stable iff:

$$\left(C_B \le \pi\cdot C_R\right) \cup \left(C_B > \max\left(\frac{\gamma}{1-R_A/C_R}, \pi\cdot C_R\right)\right) \quad (12)$$

Proof: By combining *Theorem 4* and *Theorem 5*. The complete proof is given in [22]. ■

Example: All flit rates and capacities are specified in *Kflits/sec*. Suppose L=1000 *flits/pckt*, $r_A=r_B$=100 *pckts/sec* and $R_A=R_B$=100 *Kflits/sec*. Also, B_B=16 *flits*, C_A=300, C_R=272 (namely, $C_A>C_R$ as in Case (2)). This leads to the following conditions for stability:

$$(C_B \le 4.36) \text{ OR } (C_B > 155.3)$$

The left-hand relation contradicts the necessary conditions. Hence the network is stable iff C_B>155.3. For instance, with C_B=105, queue B is unstable.

However, if C_A is reduced so that $C_A+C_B \le C_R$, i.e. $C_A \le 167$, the capacity conditions satisfy Case (1) and the network is stable. In other words, having started with an unstable network, decreasing the capacity of only one link can yield a stable network. This is the CAP phenomenon.

Note that as we increase the capacity of links A and R, we eventually obtain a stable network even for a very low capacity of link B (as long as it is strictly greater than R_B). In the example, if we want the network to be stable with C_B=105 we need to fulfill the following:

TABLE 1 SUMMARY OF WORMHOLE STABILITY CONDITIONS (IN ADDITION TO THE NECESSARY CONDITIONS)

Arbitration	Conditions	
EPRR	$C_A+C_B \le C_R$	NONE
	$C_i \ge C_R$, $C_j < C_R$ $i,j \in \{A,B\}$, $i \ne j$	If $C_j \le (B_j/L)\cdot C_R$ then $[1-(L\cdot r_i)/C_R]\cdot C_j + B_j\cdot r_i > R_j$. Otherwise NONE.
	$C_A, C_B < C_R$ and $C_A + C_B \ge C_R$	(Approximation) For $i,j \in \{A,B\}$, $i \ne j$, if $C_j > (B_j/B_i)(C_R-C_i)$ then: $\left(1-\left(t_e^i - t_f^j\right)\frac{L}{t_e^i C_R} r_i\right) C_j > R_j$ Otherwise NONE.
	$C_A, C_B \ge C_R$	NONE.
RRPF	(Approximation) There exist $0 < P_0^A, P_0^B \le 1$ that solve the equations: $P_0^A = 1-\frac{L\cdot r_A}{P_0^B\cdot C_A^e + \left(1-P_0^B\right)C_A^f}$, $P_0^B = 1-\frac{L\cdot r_B}{P_0^A\cdot C_B^e + \left(1-P_0^A\right)C_B^f}$.	
GPS	NONE.	

$(C_R \ge 6562)$ OR $(C_R > 1590)$ => $(C_R > 1590)$

Another option would be to switch to case (1) without changing C_A, by requiring that $C_R \ge 405$.

The general analysis of the symmetric case of $C_B \ge C_R$ and $C_A < C_R$ is similar to the foregoing discussion.

Case (3): $\mathbf{C_A, C_B < C_R}$ ***and*** $\mathbf{C_A + C_B \ge C_R}$**.** In this case, the router outputs flits at a slower rate than they arrive at the router, but faster than either link A or B. Consequently, packet transmission may be interrupted for either queue.

Finding the exact maximum utilization U_i for this case is hard; instead we only provide an estimate. Let us denote by t_e^i the time it takes to empty buffer $i \in \{A,B\}$, assuming that it is initially full and that while sending flits, the buffer also continuously receives new flits from node i. Since flits leave the buffer at rate C_R and new flits arrive at the router at rate C_i, the buffer emptying rate is $C_R - C_i$. If the buffer starts full, the time to empty it is

$$t_e^i = B_i/(C_R - C_i) \tag{13}$$

Similarly, t_f^i is the time to fill buffer i when it is initially empty and not being serviced. Then

$$t_f^i = B_i/(C_i) \tag{14}$$

Theorem 7: For both (i=A, j=B) and (i=B, j=A), the maximum utilization U_j of link j is

$$U_j = 1-\left(t_e^i - t_f^j\right)^+ \frac{L}{t_e^i C_R} r_i \tag{15}$$

Proof: By definition, U_j is computed when node j always has packets to send. If $t_e^i \le t_f^j$ then by the proof of *Theorem 4* queue j sees an infinite buffer, namely link j can be fully utilized. On the other hand, when $t_e^i > t_f^j$ the utilization of link j drops below 100%. For each emptying of queue i, queue j is stalled for $t_e^i - t_f^j$ seconds. The number of flits that are sent from buffer i during the time in which it is being emptied is $t_e^i \cdot C_R$ and therefore for each packet of flow i, buffer i is emptied $L/(t_e^i \cdot C_R)$ times. These packets of flow i arrive at rate r_i, hence node j is stalled for a portion of time $(t_e^i - t_f^j)^+ \cdot L/(t_e^i \cdot C_R) \cdot r_i$. The result follows. ■

Theorem 8: The network is stable iff for both (i=A, j=B) and (i=B, j=A),

$$\left(1-\left(t_e^i - t_f^j\right)\frac{L}{t_e^i C_R} r_i\right) C_j > R_j \tag{16}$$

Proof: Based on the definition that a queue is stable iff $U_B \cdot C_j > R_j$. ■

Note that this is only an approximate condition, though simulation results will show that it is a good approximation.

Case (4): $\mathbf{C_A, C_B \ge C_R}$**.** The router outputs flits at a slower rate than either links A or B.

Theorem 9: The necessary conditions in Case (4) are also sufficient for stability of the network.

Proof: It can be shown that both queues are stable, using arguments similar to the proof of *Theorem 2*. Intuitively, both links can transmit all the arrived traffic. ■

D. Stability of RRPF

The CAP phenomenon can also be shown to exist when RRPF arbitration is employed. The following analysis assumes a buffer size of one flit. As shown by simulations in Section V below, similar results are obtained for larger buffers. P_0^i denotes the probability that queue i is empty. The dependency between queues A and B is neglected. C_e^i (resp. C_f^i) is the capacity seen by node i when queue j is empty (resp. not empty). Finally, EC_i denotes the average capacity seen by node i. The round-robin arbitration can be approximated as giving equal services rates to both flows, yielding

$$C_e^i = \min\{C_i, C_R\} \tag{17}$$

$$C_f^i = \min\{C_i, \max\{C_R/2, C_R - C_j\}\} \tag{18}$$

$$EC_i = P_0^j \cdot C_e^i + (1-P_0^j) \cdot C_f^i \tag{19}$$

The average service time of a packet from queue i is approximated as L/EC_i, and by Little's law, if both queues are stable then

$$P_0^i = 1-\frac{L\cdot r_i}{EC_i} = 1-\frac{L\cdot r_i}{P_0^j\cdot C_e^i + \left(1-P_0^j\right)C_f^i} \tag{20}$$

Theorem 10: For the network to be stable, there needs to be

a solution for the following set of equations yielding $0<P_0^A, P_0^B \leq 1$:

$$P_0^A = 1 - \frac{L \cdot r_A}{P_0^B \cdot C_e^A + \left(1 - P_0^B\right) C_f^A} \quad (21)$$

$$P_0^B = 1 - \frac{L \cdot r_B}{P_0^A \cdot C_e^B + \left(1 - P_0^A\right) C_f^B} \quad (22)$$

Proof: The complete proof is given in [22]. ■

This theorem, as also confirmed by simulations, shows that the CAP phenomenon may exist when using RRPF arbitration. Note that the necessary stability conditions imply $C_e^i > R_i$ and therefore if $C_f^i > R_i$ then $C_i > P_0^j \cdot R_i + (1-P_0^j) \cdot R_i = R_i$, namely queue i is stable. Note further that if $R_A = R_B = R$ then $C_R > R_A + R_B = 2R$. Thus, $C_f^i \geq \min\{C_i, C_R/2\}$ and as noted above the network is stable. In other words, if flits arrive from all inputs at the same rate, the network is stable. This might explain why the CAP phenomenon has eluded most previous research works, since wormhole networks have typically been analyzed assuming uniform traffic

E. Stability of GPS

We now show that networks using flit-level GPS are stable. First, consider a baseline network with capacity allocation $C_A = R_A + \varepsilon$, $C_B = R_B + \varepsilon$, $C_R = R_A + R_B + 2\varepsilon$ for some small ε. Using the definition, the router output bandwidth is divided between the two flows according to their respective flit rates, namely $C_R R_A/(R_A+R_B)$ and $C_R R_B/(R_A+R_B)$, and priority is assigned dynamically according to the flit completion time in bit-by-bit GPS.

Theorem 11: The baseline flit-level GPS network is stable.

Proof: Similar to the proof of *Theorem 1*. ■

Now consider increasing either C_A, C_B or C_R but without changing the proportions that determine how the router output bandwidth is divided between the two flows.

Theorem 12: Under GPS arbitration, the necessary conditions are also sufficient for the stability of the network.

Proof: The complete proof is elaborated in [22]. Intuitively, since packet generation rates R_A, R_B have not changed (relative to the baseline network), and the respective bandwidth allocations on the router output have not decreased, the conditions of *Theorem 12* have not changed and thus the GPS network remains stable, even when either C_A, C_B or C_R are increased. ■

F. Summary of Conditions in Wormhole Switching

Table 1 summarizes the conditions for stability for the three arbitration schemes. It assumes that the necessary stability conditions are fulfilled.

IV. CAP in Store and Forward Networks

We now show that the CAP phenomenon may also happen in a store and forward network in which all packets must eventually arrive at their destinations. As shown for wormhole networks, the CAP phenomenon occurs when buffers become full, limiting the maximum utilization of a link. In store and forward networks that employ finite buffers, when buffers become full some of the packets are lost and the source nodes need to retransmit them, increasing the packet creation rates. In such cases the necessary conditions may not be sufficient for stability. In other words, the maximum effective utilization of a link cannot reach 100% in certain cases. For the sake of brevity we analyze only a specific case in this Section and illustrate the effect with simulations (Section V.B).

A. Notations and Assumptions

Time is slotted and *TS* stands for *Time-Slot*. The packet creation process is Bernoulli distributed with parameters $p_A, p_B \leq 0.5$ and $q_i = 1 - p_i$. The rates of links R and B are $C_R = 1$*pckt/TS*, $C_B = 0.5$*pckt/TS*. They fulfill the necessary stability conditions, namely $C_R \geq p_A + p_B$, and $C_B \geq p_B$. P_0^i denotes the probability that queue $i \in \{A,B\}$ is empty. We assume that all packets must eventually arrive at their destination: if a packet arrives at a full buffer and is dropped, the source retransmits the packet. Buffer size is one packet. Finally, assume that packets from A have higher priority than packets from B.

We show that there are arrival rates for which the network is stable when $C_A = 0.5$*pckt/TS* and is unstable when $C_A = 1$*pckt/TS*. Thus, one can increase capacity and consequently destabilize the network.

B. Store and Forward: Low Capacity Network

Assuming $C_A = 0.5$*pckt/TS*, we prove that the network is stable for any arrival rates $p_A, p_B \leq 0.5$.

Theorem 13: For $C_A = 0.5$*pckt/TS* every packet arriving at the router finds an empty buffer.

Proof: The proof is elaborated in [22]. Intuitively, since link R is faster than both links A and B combined, the buffer will not hold more than one flit, and a buffer of one flit is sufficient. ■

Theorem 14: Queue B is stable for $C_A = 0.5$*pckt/TS*.

Proof: Following *Theorem 13*, queue B sees an infinite buffer, i.e. its buffer is never full, and therefore it achieves the service rate of link B. That service rate fulfills the necessary stability conditions and therefore queue B is stable. ■

Theorem 15: The network is stable for $C_A = 0.5$*pckt/TS*.

Proof: Similarly, for $C_A = 0.5$*pckt/TS* queue A is stable as well and therefore the entire network is stable. ■

C. Store and Forward: High Capacity Network

Assuming $C_A = 1$*pckt/TS*, we now show that there are pairs of $p_A, p_B \leq 0.5$ for which the network is unstable.

Theorem 16: For $C_A = 1$*pckt/TS*, the probability that buffer A is full at a given *TS* is p_A.

Proof: Since link A sends one packet every TS, the state of queue A in the i^{th} *TS* is the state of buffer A in *TS* $i+1$. ■

Theorem 17: For $C_A = 1$*pckt/TS* the probability that a packet arrives and finds buffer B full is modeled by

$$p_A^2 \frac{1 - P_0^B}{q_A P_0^B + 2 - 2P_0^B} \cdot$$

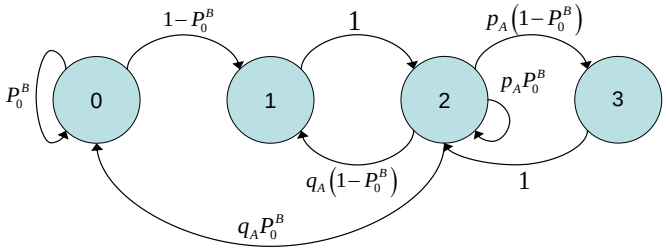

Figure 3: Markov Chain for Buffer B

Proof: Consider the Markov Chain in Figure 3 with the following states:

State 0: Buffer B is empty and there is no pending packet. This is also the initial state.

State 1: Buffer B is empty but a new packet will arrive in the next *TS*.

State 2: Buffer B stores a packet and no packet will arrive in the next *TS* (there is no pending packet).

State 3: Buffer B stores a packet and a new packet will arrive in the next *TS*.

A detailed proof and explanation of the transitions is given in [22]. Here we only explain the transition from *state 2* to *state 3* as an example. That transition happens if the packet in buffer B is not serviced and queue B is not empty. Since the probability that buffer B is not serviced is the same as having a packet in buffer A (i.e. it is p_A, according to *Theorem 16*), the transition happens with probability $p_A \cdot (1-P_0^B)$.

Note that in the Markov chain we use P_0^B. We recognize that this is sometimes inaccurate and we should have used the probability that queue B is empty given that there was a departure two or more *TS*s previously. However, since the latter probability is unknown, we use P_0^B as an approximation. As demonstrated by the simulations, this approximation yields satisfactory results [22]. ■

The following result is proved in [22].

Theorem 18: For C_A=1*pckt/TS* queue B is stable whenever $p_B+p_A\pi_3<0.5$.

We can now establish the CAP phenomenon in store and forward networks.

Theorem 19: There exist arrival rates p_A,p_B for which the network with C_A=0.5*pckt/TS* is stable but for C_A=1*pckt/TS* it is unstable. Therefore, the CAP phenomenon also appears in store and forward networks, since one can add capacity and destabilize the network.

Proof: By example. Consider $p_A=p_B=0.45$. *Theorem 15* shows that the network with C_A=0.5*pckt/TS* is stable for these arrival rates, and *Theorem 18* shows that the condition for stability when C_A=1*pckt/TS* is $p_B+p_A\pi_3<0.5$, which in this case reduces to $\pi_3<0.11$. It can also be seen that π_3 is decreasing when P_0^B increases. Clearly, $P_0^B \le 1-2p_B=0.1$. Therefore $\pi_3 \ge 0.23 > 0.11$ and the network is unstable. ■

Thus, we have proven that one can destabilize a store and forward network by adding capacity. We can also identify the conditions for stability when C_A=1*pckt/TS*.

Theorem 20: When C_A=1*pckt/TS*, queue B is stable iff $p_B + \frac{1}{2}p_A^2 < \frac{1}{2}$.

Proof: Intuitively, if we consider queue B to be unstable, namely take P_0^B=0 in the Markov chain in Figure 3, then it is easy to see that we get π_2=0.5, and $\pi_3=0.5 \cdot p_A$. As already mentioned the probability for a packet to arrive and to find buffer B full is $p_A\pi_3$, and since all packets must arrive at their destinations, queue B will need to resend those packets. Therefore the probability that a packet will be added to queue B is $p_B+p_A\pi_3$, and if it smaller than 0.5 then indeed queue B is stable. The complete proof is elaborated in [22]. ■

V. Simulations Results

We now present simulations results that demonstrate the existence of the CAP phenomenon and confirm the foregoing analysis. Additional interesting simulations can be found in [22].

A. Wormhole Networks

An OPNET-based wormhole simulator [19][28][29] was used to verify our results. Each run simulates 1000 seconds, divided into 20 intervals. To show stability or instability of a network we consider the average End to End (ETE) delay of packets on each interval. The ETE delay is defined as the time elapsed between the creation of a packet and the arrival of the last flit of the packet at the destination. In a stable network, ETE delay is bounded. In an unstable network ETE typically increases near-linearly.

The simulation uses L=1000 *flit/pckt*, N=10 *bit/flit*, r_A=500 *pckt/sec*, r_B=100 *pckt/sec*, implying R_A=500 *Kflit/sec* and R_B=100 *Kflit/sec*. In addition B_i=16 *flit* and C_{RC}=636 *Kflit/sec*.

Figure 4 shows wormhole configurations in which the mathematical analysis yields a stable or unstable network (colored gray or black, respectively). Figure 5 zooms in on areas where the necessary stability conditions are fulfilled. Crosses and squares represent simulated stable and unstable configurations, respectively. As can be seen, our analysis predicts the stability condition well, and the CAP phenomenon appears for both EPRR and RRPF but not for GPS.

One interesting note for EPRR is that for stability of the network when $C_A \ge C_R$=636 *Kflit/sec* we need C_B>450 *Kflit/sec*, when in fact link B should only transmit R_B=100 *Kflit/sec*. Furthermore, one can see that for EPRR there is a narrow tube in the stability region. In this case, any small deviation from the optimal C_A, either upwards or downwards, will make the network unstable. Note also that for RRPF when C_B increases the network becomes unstable due to queue A [22]. This is characteristic of the stability condition for RRPF: only the node with the higher packet generation rate may become unstable as a result of increased link capacity. In contrast, for EPRR either one of the queues may become unstable as a result of increased capacity.

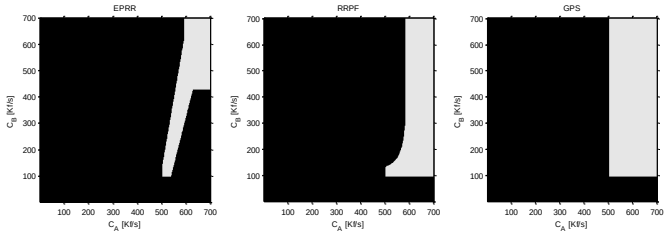

Figure 4: Unstable (black) and stable (gray) configurations according to wormhole analysis when R_A=500*Kflit/sec*, and R_B=100*Kflit/sec*. The figures refer (from left to right) to EPRR, RRPF and GPS

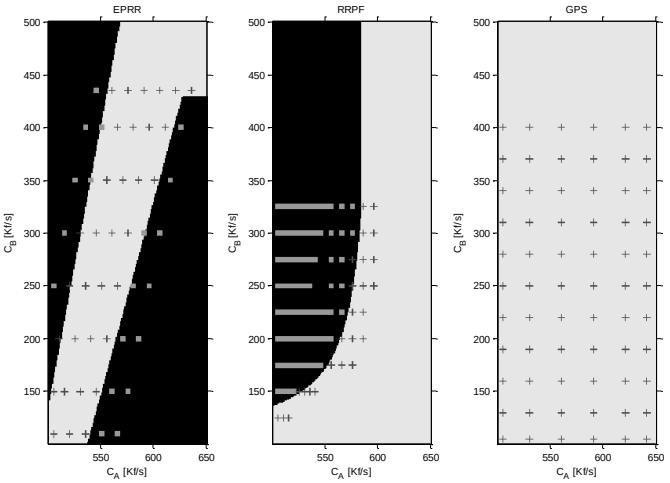

Figure 5: Wormhole results for R_A=500*Kflit/sec*, R_B=100*Kflit/sec*. Unstable (black) and stable (gray) configurations according to analysis are compared to unstable (squares) and stable (crosses) configurations according to simulations. The figures refer (from left to right) to EPRR, RRPF and GPS

B. *Store and Forward Networks*

Figure 6 presents the simulation results for the analysis of Section IV. We used Matlab for those simulations. Each square represents one simulation run, where black and gray squares represent stable and unstable simulations, respectively. For C_A=1*pckt/TS* the black line represents the limit between stable (below the line) and unstable (above) configurations according to *Theorem 20*. It can be seen that the analysis is quite accurate. For C_A=0.5*pckt/TS* the network is always stable, whereas for C_A=1*pckt/TS* it is not always stable.

Figure 7 shows four additional simulations, performed using OPNET. The following arbitrations have been used in the router:

Priority based: The router tries to serve the higher priority traffic before the other traffic. Low priority packets are serviced only if there are no high priority packets.

Round Robin: When both buffers are not empty the router transmits one packet from each queue in a round robin fashion.

Exhaustive: Once a buffer is being served, it is served until it becomes empty. Table 2 summarizes the parameters for each simulation. It can be seen that the CAP phenomenon appears in all configurations and the comments made above about wormhole simulations apply here as well.

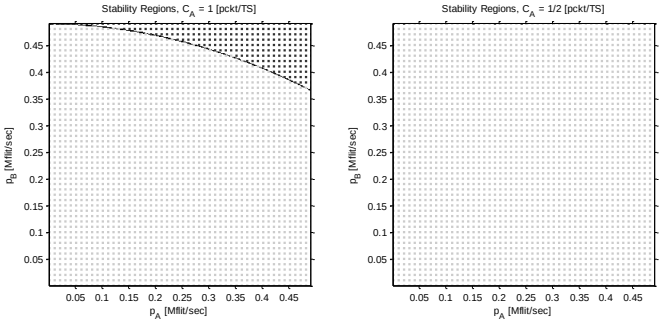

Figure 6: Store and Forward simulation results. Gray/Black squares (dots) represent stable/unstable simulations. On the left hand figure, analysis suggests that every configuration above the black line is unstable.

TABLE 2 SIMULATIONS PARAMETERS FOR STORE AND FORWARD

Parameter	**(a)**	**(b)**	**(c)**	**(d)**
Creation Process	Poisson	Poisson	Poisson	Poisson
Poisson Parameter	A: 100 B: 100	A: 100 B: 100	A: 500 B: 100	A: 500 B: 100
Packet Length [bits]	10^4	10^4	10^4	10^4
Bits created per second	A: 1M B: 1M	A: 1M B: 1M	A: 5M B: 1M	A: 5M B: 1M
Buffer Size [packets]	4	3	3	3
Arbitration	Priority (A has higher priority)	Exhaustive	Round-Robin	Exhaustive

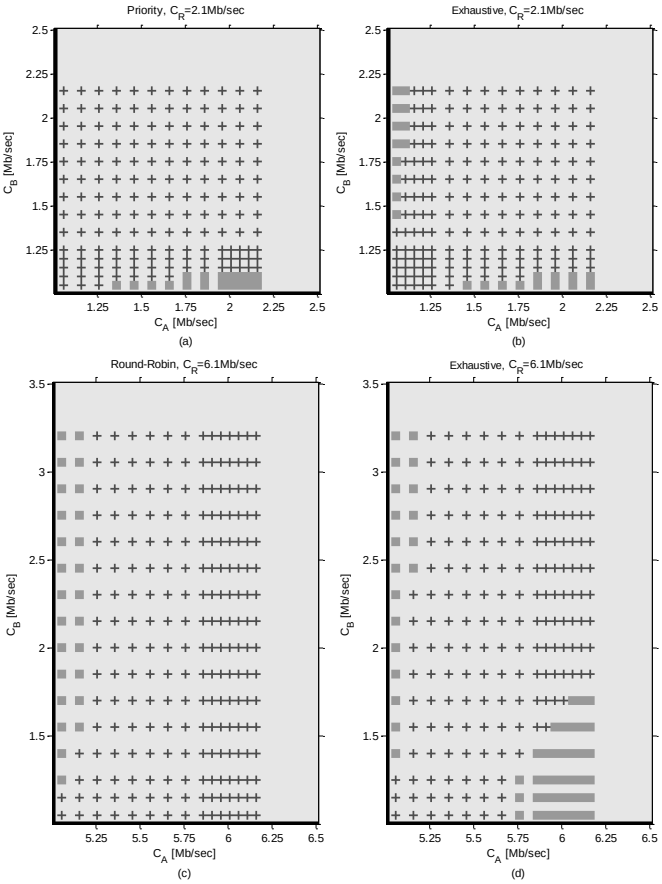

Figure 7: Simulation results for Store and Forward. Parameters can be seen in Table 2. Crosses/squares represent stable/unstable configurations according to simulations. Adding capacity may destabilize the network.

VI. Conclusions

In this paper, we introduced the Capacity Allocation Paradox (CAP), and showed how a stable finite-buffer network can become unstable when a link capacity is increased. We demonstrated it in a basic 2x1 network topology, and showed how it applies to fluid, wormhole and packet-switched networks, using various common scheduling algorithms.

In addition, we proved that in some stable networks, the CAP phenomenon might be so strong that increasing a link capacity to infinity makes the network unstable, and cannot restore stability anymore. Thus *CAP prevents any natural capacity allocation algorithm*, because there is no point in arbitrarily increasing some link capacity until the stable point is found.

Further, we proved that in some stable networks, any small deviation from an optimal link capacity, *both* upwards or downwards, will make the network unstable. This also makes capacity allocation extremely hard, because the stable region might be too small to find by optimization algorithms.

Finally, we showed that network designers can actually find solutions to the CAP phenomenon. For instance, they can use flit-level GPS arbitration, or force router output links to have larger capacity than the sum of the input link capacities. However, these solutions are not necessarily practical, emphasizing how much the CAP phenomenon might be hard to avoid in capacity allocation algorithms.

Acknowledgment

This work was partly supported by the European Research Council Starting Grant n^o 210389, the Alon Fellowship, and the ATS-WD Career Development Chair.

References

[1] L. Kleinrock, *Queueing Systems*, New York: Wiley, 1975.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York: Wiley, 1991.

[3] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multi-hop wireless networks," *IEEE Infocom*, Phoenix, Arizona, USA, Apr. 2008.

[4] P. R. Kumar and S. P. Meyn, "Stability of queuing networks and scheduling policies," *IEEE Trans. Automatic Control*, vol. 40, pp. 251-260, Feb. 1995

[5] P. Giaccone, E. Leonardi, and D. Shah, "Throughput region of finite-buffered networks," *IEEE Transactions on Parallel and Distributed Systems*, vol.18, no.2, Feb. 2007.

[6] R. Urgaonkar and M. J. Neely, "Capacity region, minimum energy, and delay for a mobile ad-hoc network," *WiOpt*, Apr. 2006.

[7] W. J. Dally, "Virtual-channel flow control," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194-205, Mar. 1992.

[8] I. Cohen, O. Rottenstreich and I. Keslassy, "Statistical Approach to NoC design," *ACM/IEEE NoCS '08*, Newcastle, UK, Apr. 2008.

[9] M. J. Neely, E. Modiano, and C. E. Rohrs, "Power allocation and routing in multi-beam satellites with time varying channels," *IEEE Transactions on Networking*, vol. 11, no. 1, pp. 138-152, Feb. 2003.

[10] D. Braess, "On a paradox of traffic planning," *Transportation Science*, vol. 39, no. 4, pp. 446–450, 2005.

[11] J. E. Cohen, and F. P. Kelly, "A paradox of congestion in a queuing network," *J. Appl. Prob.*, vol. 27, pp. 730–734, 1990.

[12] B. Calvert, W. Solomon, and I. Ziedins. "Braess's paradox in a queueing network with state-dependent routing," *Journal of Applied Probability*, vol. 34, pp. 134–154, 1997.

[13] T. Roughgarden, "Designing Networks for selfish users is hard," *42nd IEEE symposium on Foundations of Computer Science*, p.472, Oct. 2001.

[14] J. G. Dai, J. J. Hasenbein, and J. H. Vande Vate, "Stability of a three-station fluid network," *Queueing Systems: Theory and Applications*, vol. 33, no. 4, pp. 293-325, 1999.

[15] P. R. Kumar and T. I. Seidman, "Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems," *IEEE Trans. Automat. Contr.*, vol. 35, pp. 289-298, Mar. 1990.

[16] J. G. Dai and G. Weiss, "Stability and instability of fluid models for re-entrant lines," *Math. Oper. Res.*, vol. 21, pp. 115-134, 1996.

[17] M. Bramson, "Instability of FIFO queuing networks," *Ann. Appl. Probab.*, vol. 4, no. 2, pp. 414-431, 1994.

[18] T.I. Seidman, "'First come, first served' can be unstable!," *IEEE. Trans. Automat. Control*, vol. 39, pp. 2166-2171, 1994.

[19] A. Baron, I. Walter, R. Ginosar, I. Keslassy, and O. Lapid, "Benchmarking SpaceWire Networks," *Int. SpaceWire Conf.*, 2007.

[20] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny. "Efficient link capacity and QoS design for network-on-chip," *DATE*, 2006.

[21] A. Baron, I. Walter, I. Cidon, R. Ginosar, I. Keslassy, and O. Lapid, "SpaceWire Hot Modules," *Int. SpaceWire Conf.*, 2007.

[22] A. Baron, R. Ginosar, and I. Keslassy, "The capacity allocation paradox," *Technical Report TR08-02*, Comnet, Technion, Israel. Available: http://comnet.technion.ac.il/~isaac/papers.html

[23] R. Loynes, "The stability of a queue with non-independent inter-arrival and service times," *Proc. Cambr. Phil. Soc.*, vol. 58, no. 3, pp. 497–520, 1962.

[24] H. Sethu, H. Shi, S.S. Kanhere, and A.B. Parekh, "A round-robin scheduling strategy for reduced delays in wormhole switches with virtual lanes," *Proc. Int. Conf. Comm. in Computing*, June 2000.

[25] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, June 1996.

[26] S. S. Kanhere, H. Sethu, and A. B. Parekh, "Fair and efficient packet scheduling using elastic round robin," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 324–336, March 2002.

[27] A.K. Parekh and R. G. Gallager, "A Generalized Processor Sharing approach to flow control in integrated services networks: The single node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344-357, June 1993.

[28] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "QNoC: QoS architecture and design process for networks on chip," *Journal of Systems Architecture*, vol. 50, pp. 105-128, 2004.

[29] OPNET Modeler, www.opnet.com.