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.

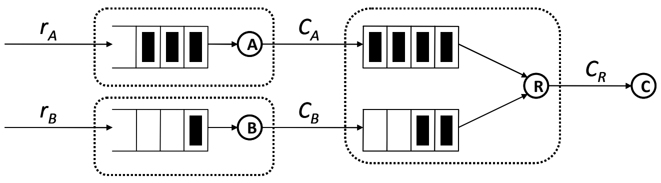

Figure 1: Simple 2×1 network

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 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], spacecraft networks [19]. 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 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.

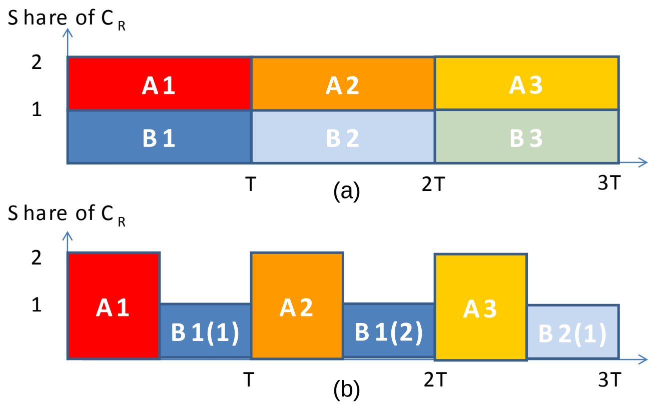

Figure 2: Sharing of Link R, using r_A=r_B=1, C_B=1, C_R=2.
(a) C_A =1, stable network (b) C_A =2, unstable network.

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 flits cannot be forwarded from a given node, 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 $R_i = r_i \cdot L$. Let link i denote the link that is 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 $B_A, B_B \ll L$ (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 $U_i \cdot C_i > R_i$ [27].

We consider four common work conserving arbitrations as follows:

1) *Exhaustive Packet Round Robin (EPRR)* [22]: 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) *Priority* [21]: The router gives higher priority to packets from one of the nodes. Thus if the buffer with the higher priority is not empty it gets service, otherwise the router serves the buffer with the lower priority.
3) *Round Robin Per Flit (RRPF)* [23][24]: The router transmits one flit from one of the buffers and then serves the other buffer, interleaving flits from the two buffers.
4) *General Processor Sharing (GPS)* [26], 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 the EPRR, RRPF and Priority arbitrations, whereas it does not occur when using GPS.

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:

$$C_A > R_A \quad (7)$$

$$C_B > R_B \quad (8)$$

$$C_R > R_A + R_B \tag{9}$$

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 two buffers together will hold no more than one flit, where only flits that are completely in the buffer are counted.

Proof: Instead of considering two buffers we consider one faster buffer which both links A and B write to it bit by bit. Since the number of bits in a flits is N, to prove the theorem it is sufficient to show that the number of bits in the buffer cannot exceed 2N-1. Once the number of bits in the buffer reaches 2N-1, it is guaranteed that there is one whole flit in the buffer and thus the router starts to transmit it. Since $C_A+C_B \le C_R$, the arrival rate to the buffer is smaller than the departure rate and therefore the number of bits in the buffer starts to decrease. Therefore the buffer will hold no more than 2N-1 bits and the proof is complete. ■

Theorem 2: When $C_A + C_B \le C_R$, the necessary stability conditions are also sufficient.

Proof: From Theorem 1 we know that the buffers will hold no more than one flit, and therefore the queue of node *i* gets service 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 3: 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 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:

$$\left(1-\frac{R_B}{C_R}\right)\cdot\min\{C_A, C_R\} > R_A \underset{C_A \ge C_R}{\Leftrightarrow} \left(1-\frac{R_B}{C_R}\right)\cdot C_R > R_A$$

$$\Leftrightarrow \underbrace{C_R > R_A + R_B}_{\text{Necessary Condition}} \qquad ■$$

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 4: 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 5: Denote $x^+=\max(x,0)$, then the maximum utilization of link B is

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

Proof: By definition, U_B is computed when queue B is always non-empty. Theorem 4 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 6: Queue B is stable iff

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

Proof: By definition, queue B is stable iff $U_B \cdot C_B > R_B$. Using the result of Theorem 5, 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 7: In Case (2), queue B is stable iff:

$$\left(C_B \le \pi\cdot C_R\right) \text{ OR } \left(C_B > \frac{\gamma}{1-R_A/C_R}\right) \tag{12}$$

Proof: From Theorem 6 we get that queue B is stable iff

$$\left(C_B \le \pi\cdot C_R \cap C_B > R_B\right) \text{ OR } \left(C_B > \pi\cdot C_R \cap C_B > \frac{\gamma}{1-R_A/C_R}\right)$$

When the necessary conditions are fulfilled this becomes

$$\left(C_B \le \pi\cdot C_R\right) \text{ OR } \left(C_B > \pi\cdot C_R \cap C_B > \frac{\gamma}{1-R_A/C_R}\right)$$

Therefore if the necessary conditions are fulfilled and $C_B \le \pi\cdot C_R$ the network is stable, otherwise $C_B > \pi\cdot C_R$ and for stability we demand $C_B > \dfrac{\gamma}{1-R_A/C_R}$. This completes the proof. ■

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)). From Equation (12) we get the following conditions for stability:

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

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 $C_A + C_B \leq C_R$, i.e. $C_A \leq 167$, 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:

$$(C_R \geq 6562) \text{ OR } (C_R > 1590) \Rightarrow (C_R > 1590)$$

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

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

***Case (3):* $\mathbf{C_A, C_B < C_R}$ *and* $\mathbf{C_A + C_B \geq C_R}$.** 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

$$t_e^i = B_i/(C_R - C_i) \qquad (13)$$

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

$$t_f^i = B_i/C_i \qquad (14)$$

Theorem 8: For both (i=A, j=B) and (i=B, j=A), the maximum utilization U_j of link j is

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

Proof: By definition, U_j is computed when node j always has packets to send. If $t_e^i \leq t_f^j$ then by the proof of Theorem 5 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 9: The network is stable iff for both (i=A, j=B) and (i=B, j=A),

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

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):* $\boldsymbol{C_A, C_B \geq C_R}$.** The router outputs flits at a slower rate than either links A or B.

Theorem 10: The necessary conditions in Case (4) are also sufficient for stability of the network.

Proof: From Theorem 3 queue A is stable when $C_A \geq C_R$. Obviously when $C_B \geq C_R$, queue B is stable. Therefore both queues are stable and so is the network. ■

D. *Stability of Priority*

Assume that the router uses Priority arbitration and without loss of generality assume packets from A have higher priority than packets from B. First we prove that the necessary stability conditions are sufficient for queue A. Then we break the discussion of the stability conditions of queue B into several cases in the same manner we did for EPRR, and show that for some capacity allocations queue B is stable, while increasing the capacity of link A may destabilize the network.

Theorem 11: The necessary stability conditions are also sufficient for the stability of queue A.

Proof: Since packets from A have higher priority, the capacity seen by node A is given by $min\{C_A, C_R\}$. Therefore the conditions $C_A > R_A$ and $C_R > R_A$ are sufficient for the stability of queue A and thus the proof is complete. ■

From now on we deal only with the stability of queue B. In contrast with EPRR where we know buffer B gets service after every packet that is transmitted from buffer A, here buffer B gets service only once queue A becomes empty. This difference makes it very difficult to find the exact stability conditions for queue B. Therefore for cases (2a) and (3) below we only find sufficient conditions for instability of queue B. However we give approximated stability conditions for the case where the creation process of packets in queue A is Poisson. We assume the *necessary stability conditions are fulfilled.*

Case (1): $\mathbf{C_A + C_B \leq C_R}$.

Theorem 12: For $C_A + C_B \leq C_R$, the necessary stability conditions are also sufficient for the stability of queue B.

Proof: Note that for this case both buffers will hold no more than one flit (the proof is the same as the one of Theorem 1). The proof is then completed similar to the proof of Theorem 2. ■

Case (2a): $\mathbf{C_A \geq C_R}$ *and* $\mathbf{C_B < C_R}$. In this case, the router output link transmits flits slower than higher priority link A but faster than lower priority link B. We show the necessary stability conditions may not be sufficient for queue B.

Theorem 13: Queue B is unstable whenever

$$C_B < \frac{R_B}{1 - r_A\left(\frac{L}{C_R} - \frac{B_B}{C_B}\right)^+} \tag{17}$$

Proof: Note that for case (2a) and when packets from A have higher priority, the Priority arbitration is almost the same as EPPR. The only difference is that the transmission of packets from buffer A doesn't end when the transmission of a packet end but when buffer A becomes empty, which necessarily implies that queue A becomes empty. Therefore in Priority as opposed to EPRR link B will be able to fill its buffer only for some of the packets that node A transmits, and the maximum utilization of link B is even lower than we had in EPRR. Therefore as in Theorem 4-Theorem 6, we can show that Equation (17) is sufficient for the instability of queue B (however it is not necessary). ■

In the last proof we assume buffer B empties after every transmission of a packet from node A which is obviously not true in many cases. We denote by EQ_A the expected number of flits in queue A. The following theorem estimates the stability conditions for queue B.

Theorem 14: For case (2a) the stability region queue B is modeled by

$$C_B > \frac{R_B}{1 - \frac{R_A}{EQ_A}\left(\frac{EQ_A}{C_R} - \frac{B_B}{C_B}\right)^+} \tag{18}$$

Proof: The proof relies on an approximation of U_B. Once buffer A is not empty, the router will transmit about EQ_A flits from it before it gets empty and only then buffer B gets service. The time it takes to send EQ_A flits on link R is EQ_A/C_R, and the time it takes to fill buffer B is B_B/C_B. Therefore if $EQ_A/C_R \le B_B/C_B$, then the maximum utilization of link B may be 100%. Otherwise, the maximum utilization cannot be 100% and for every EQ_A flits that are transmitted from buffer A link B will not work for EQ_A/C_R-B_B/C_B. Thus the part of the time link B cannot work is approximately $(R_A/EQ_A)(EQ_A/C_R\text{-}B_B/C_B)^+$. From the definition of stability the theorem follows. ■

The problem with the last theorem is that it requires the knowledge of EQ_A which sometimes is unknown. We cannot tell EQ_A for the general case, but when the creation process of packets in queue A is Poisson, then since packets from A have higher priority it can be modeled as an M/D/1 queue and we can find EQ_A using the Pollaczek-Khinchine formula and Little's law. The service rate of packets from A is $\mu_A = min\{C_A, C_R\}/L = C_R/L$. According to the Pollaczek-Khinchine formula the average waiting time of a packet in queue A is

$$EW = \frac{r_A}{2\mu_A(\mu_A - r_A)} \tag{19}$$

And according to little's law $EQ_A = L \cdot r_A(EW + 1/\mu_A)$, which yield

$$EQ_A = L\frac{r_A}{\mu_A}\frac{2\mu_A - r_A}{2(\mu_A - r_A)} \tag{20}$$

Since in our simulations the creation process is Poisson we use Equation (20) to verify the results, section V.A shows that the model is close to simulation results.

Case (2b): $C_A < C_R$ *and* $C_B \ge C_R$. *In this case, the router output link transmits flits faster than higher priority link A but slower than lower priority link* B. We show the necessary stability conditions are also sufficient for queue B.

Theorem 15: For case (2b) necessary stability conditions are also sufficient for queue B.

Proof: Exactly as the proof of Theorem 3. ■

Case (3): **$C_A, C_B < C_R$ *and* $C_A + C_B \ge C_R$.**

As in case (3) of EPRR it is very hard to find the exact maximum utilization of link B, and the fact that buffer B gets service only when buffer A is empty makes it even harder. Therefore to find sufficient conditions for instability we use more or less the same approach as in case (3) of EPRR, and assume that between every two consecutive packets that are sent from A, buffer B is getting emptied and then refilled.

As earlier, t_f^i denotes the time to fill buffer i when it is initially empty and not being serviced. As before

$$t_f^i = B_i/C_i \tag{21}$$

We denote by t_p^i the time it takes to transmit a packet on link i, obviously

$$t_p^i = L/C_i \tag{22}$$

Theorem 16: Queue B is unstable whenever

$$\left[1 - \left(t_p^A - t_f^B - \left(t_p^A - t_p^R\right)\frac{C_R}{C_B}\right)^+ r_A\right] C_B \le R_B \tag{23}$$

Proof: The amount of time it takes link R to transmit a packet is t_p^R, but since link A is slower, it will take link R $t_p^A > t_p^R$ to transmit the packet from A. Therefore, for every packet from A, buffer B gets service for t_p^A-t_p^R. During this time link R sends $(t_p^A\text{-}t_p^R)C_R$ flits from buffer B, and link B can send other flits instead. Therefore for every packet sent from A, link B can work for $[(t_p^A\text{-}t_p^R)C_R]/C_B$. In addition, to find the maximum utilization possible, for every packet sent from A link B can also fill buffer B. Combining it all together we get that the maximum utilization of link B fulfills

$$U_B \le 1 - \left(t_p^A - t_f^B - \left(t_p^A - t_p^R\right)\frac{C_R}{C_B}\right)^+ r_A$$

By definition of stability queue B is unstable iff $U_B C_B \le R_B$ which yield Equation (23) and the proof is complete. ■

Again, the last theorem only presents a sufficient condition for instability since we assume buffer B is emptied after every transmission of a packet from node A. As in case (2a) we want to give a more accurate condition for stability.

Theorem 17: For case (3), the stability region queue B is modeled by

$$\left[1 - \left(t_p^A - \frac{L \cdot t_f^B}{EQ_A} - \left(t_p^A - t_p^R\right)\frac{C_R}{C_B}\right)^+ r_A\right] C_B > R_B \tag{24}$$

Proof: The proof is similar to the previous one with the difference that buffer B empties only once every EQ_A flits that are sent from buffer A. ■

Again, given a Poisson packet creation process we can model queue A as an M/D/1 queue with service rate $\mu_A=min\{C_A,C_R\}/L=C_A/L$, and the expression for EQ_A is given in Equation (20).

Case (4): $C_A,C_B \geq C_R$.

Theorem 18: The necessary conditions in Case (4) are also sufficient for stability of the network.

Proof: It can be shown that queue B is stable, using arguments similar to the proof of Theorem 3. ■

E. 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 – although similar results are obtained for larger buffers as illustrated in the simulation results of Section V. The dependency between queues A and B is neglected and thus this is an approximated model. However, we note that if one of the queues is not stable then there is no dependency between the two queues. Therefore when we model the stability of the network it is reasonable to neglect the dependency between queues A and B.

We use P_0^i to denote the probability that queue i is empty. For (i,j)=(A,B) or (i,j)=(B,A), 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

$$C_e^i = \min\{C_i,C_R\} \quad (25)$$

$$C_f^i = \min\{C_i, \max\{C_R/2, C_R - C_j\}\} \quad (26)$$

$$EC_i = P_0^j \cdot C_e^i + (1-P_0^j) \cdot C_f^i \quad (27)$$

The creation rate of flits in queue i is given by $L\cdot r_i=R_i$ and by Little's law. If both queues are stable then

$$P_0^i = 1-\frac{R_i}{EC_i} = 1-\frac{R_i}{P_0^j\cdot C_e^i+\left(1-P_0^j\right)C_f^i} \quad (28)$$

Theorem 19: When the dependency between queues A and B is neglected, the network is stable iff there is a solution for the following set of equations yielding $0<P_0^A, P_0^B \leq 1$:

$$P_0^A = 1-\frac{R_A}{P_0^B\cdot C_e^A+\left(1-P_0^B\right)C_f^A} \quad (29)$$

$$P_0^B = 1-\frac{R_B}{P_0^A\cdot C_e^B+\left(1-P_0^A\right)C_f^B} \quad (30)$$

Proof: Assume the network is stable and there is no solution for Equations (28) and (29) yielding $0<P_0^A, P_0^B \leq 1$. Since $P_0^i \leq 1$ there must exist $P_0^i \leq 0$ for i=A or i=B (or both), therefore:

$$0 \geq P_0^i = 1-\frac{R_i}{EC_i} \Leftrightarrow EC_i \leq R_i$$

which means that the network is not stable, in contradiction with our assumption.

Now assume there is a solution to the set of equations yielding $0<P_0^A, P_0^B \leq 1$, and that the network is unstable. Since there exists a solution, $0< P_0^i \leq 1$ for both i=A and i=B, therefore,

$$0 < P_0^i = 1-\frac{R_i}{EC_i} \Leftrightarrow EC_i > R_i$$

Thus the network is stable in contradiction with our assumption. By that the proof is complete. ■

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, $C_f^i \geq \min\{C_i,C_R/2\} \geq R_i$ 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.

F. Stability of GPS

We now show that networks using flit-level GPS are stable. Using the definition, the router output bandwidth is divided between the two flows according to their respective flit rates, namely $C_RR_A/(R_A+R_B)$ and $C_RR_B/(R_A+R_B)$, and priority is assigned dynamically according to the flit completion time in bit-by-bit GPS.

To show that the necessary conditions are sufficient for stability when GPS is used we consider a baseline network which fulfills only the necessary condition, namely $C_A=R_A+\varepsilon$, $C_B=R_B+\varepsilon$, $C_R=R_A+R_B+2\varepsilon$ for some small ε. We show this network is stable and that adding capacity to this network cannot destabilize it.

Theorem 20: For the baseline flit-level GPS network all links can be fully utilized, and the network is stable.

Proof: As in Theorem 1 we can show that the buffers will hold no more than one flit and thus all links can be fully utilized. The proof of stability is similar to the proof of Theorem 2. ■

We now show that increasing either C_A, C_B or C_R in respect with the baseline network but without changing the proportions that determine how the router output bandwidth is divided between the two flows cannot destabilizes the network.

Theorem 21: Increasing the capacity of link R in respect to the baseline capacity allocation cannot destabilize the network.

Proof: When we increase C_R there still exists $C_R \geq C_A+C_B$ and therefore the buffers will hold no more than one flit (Theorem 1), link A and B can be fully utilized, and obviously the network is stable (Theorem 2). ■

Theorem 22: Increasing C_A or C_B in respect with the baseline capacity allocation cannot destabilize the network.

Proof: We prove the theorem for increasing C_A, the proof for increasing C_B is obviously similar. Consider the stable baseline network, which according to Theorem 20 is stable, and increase C_A. The capacity that queue B sees is limited by C_B which is still $R_B+\varepsilon$, and by the output of buffer B. In the baseline network buffer B got a share of $C_RR_B/(R_A+R_B)$ on link

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

Arbitration	Conditions	
EPRR	$C_A+C_B \le C_R$	NONE
	$C_A \ge C_R$, $C_B < C_R$	$[1-(L \cdot r_A)/C_R] \cdot C_B + B_B \cdot r_A > R_B$
	$C_A < C_R$, $C_B \ge C_R$	$[1-(L \cdot r_B)/C_R] \cdot C_A + B_A \cdot r_B > R_A$
	$C_A, C_B < C_R$ and $C_A + C_B \ge C_R$	For both (i=A, j=B) and (i=B, j=A): $\left(1-\left(t_e^i - t_f^j\right)^+ \frac{L}{t_e^i C_R} r_i\right) C_j > R_j$ (model)
	$C_A, C_B \ge C_R$	NONE
Priority (when A has higher priority)	$C_A+C_B \le C_R$	NONE
	$C_A \ge C_R$, $C_B < C_R$	$\left[1-\frac{R_A}{EQ_A}\left(\frac{EQ_A}{C_R}-\frac{B_B}{C_B}\right)^+\right] C_B > R_B$ (model)
	$C_A < C_R$, $C_B \ge C_R$	NONE
	$C_A, C_B < C_R$ and $C_A + C_B \ge C_R$	$\left[1-\left(t_{p,A}-\frac{L \cdot t_f^B}{EQ_A}-\left(t_{p,A}-t_{p,R}\right)\frac{C_R}{C_B}\right)^+ r_A\right] C_B > R_B$ (model)
	$C_A, C_B \ge C_R$	NONE
RRPF	There exist $0 < P_0^A, P_0^B \le 1$ that solve the equations: $P_0^A = 1-\frac{R_A}{P_0^B \cdot C_A^e + \left(1-P_0^B\right) C_A^f}$, $P_0^B = 1-\frac{R_B}{P_0^A \cdot C_B^e + \left(1-P_0^A\right) C_B^f}$ (model)	
GPS	NONE	

R and since none of these parameters changed queue B is still stable, and even further, buffer B will hold no more than one flit. Obviously we can use the same arguments for queue A and show it is stable. However, note that if we increase C_A so that $C_A+C_B>C_R$ link A cannot be fully utilized because buffer A may get full. Still, the capacity seen by buffer A doesn't change and therefore the maximum utilization of link A fulfills $U_A C_A \ge C_R R_A/(R_A+R_B) > R_A$. Thus, queue A is stable. ■

Theorem 23: Under GPS arbitration, the necessary conditions are also sufficient for the stability of the network.

Proof: Consider some capacity allocation $\tilde{C}_A, \tilde{C}_B$ and $\tilde{C}_R$ which fulfills necessary conditions. We start with the baseline configuration and increase C_R to $\tilde{C}_R$, from Theorem 21 we know that the network remains stable. Now we increase C_A to $\tilde{C}_A$ and using the same arguments as in the proof of Theorem 22 we can see it is stable. Finally we increase C_B to $\tilde{C}_B$ and show the network remains stable. If link *i* can be fully utilized then obviously queue *i* remains stable. On the other hand, if link *i* cannot be fully utilized then its utilization is limited by the capacity that buffer *i* sees and since C_R, R_A, and R_B were not changed, $U_i C_i \ge C_R R_i/(R_A+R_B) > R_i$, thus the network is stable. Therefore the necessary conditions are also sufficient for the stability of the network. ■

To summarize GPS arbitration, we saw that a baseline network which fulfills only the minimum capacity demands is stable and that adding capacity to this network cannot destabilizes the network and thus the necessary conditions are also sufficient for the stability of the network.

G. 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 it using 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 $p_A, p_B \le 0.5$ 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 $C_R \ge p_A + p_B$, and $C_B \ge p_B$. P_j^i denotes the probability that queue $i \in \{A,B\}$ has *j* packets. 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. Also assume that packets from A have higher priority than packets from B. Finally, as we shall see in the proof of Theorem 26 the results we obtain are approximated, although simulations show that they are very close to the real conditions. We define queue *i* to be stable iff its service rate is greater than its arrival rate.

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.

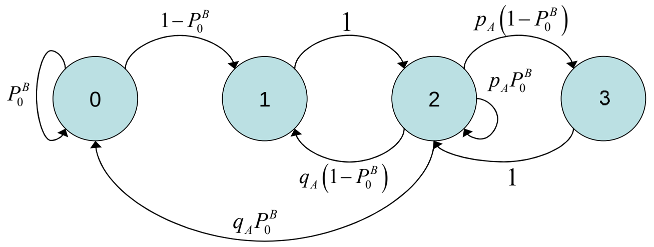

Figure 3: Markov Chain for Buffer B

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 $p_A,p_B \le 0.5$.

Theorem 24: For C_A=0.5*pckt/TS* every packet arriving at the router finds an empty buffer.

Proof: Since packets from A have higher priority, and since it takes two TSs to transmit a packet on link A and only one TS on link C it is obvious that every packet that arrives from A sees an empty buffer. Now we deal with buffer B. Let's assume that at TS #1 a packet from B arrived to the router. The next packet from B will arrive to the router no sooner than TS #3 and we need to prove it will find an empty buffer. There are two options; first one is that there is no packet in buffer A in TS #1. In this case buffer B will get service and obviously at TS #3 the buffer will be empty. The second option is that a packet from A arrived to the router at TS #1 (with the first packet from B). In this case the packet in buffer A will get service during TS #2 and the packet in buffer B will need to wait. The next packet from A cannot arrive before TS #3 since it takes two TS's to transmit a packet on link A. Therefore in TS #3 the packet in buffer B will get service and the new packet that arrived from B will find an empty buffer. This completes the proof. ■

Theorem 25: The network is stable for C_A=0.5*pckt/TS.*

Proof: Following Theorem 24, queues A and B sees an infinite buffer, i.e. their buffer is never full, and therefore both achieve the service rates of links A and B respectively. That service rate is greater than the arrival rate and therefore both queues are stable and so is the entire network. ■

C. Store and Forward: High Capacity Network

Assuming C_A=1*pckt/TS*, we now show that there are pairs of $p_A,p_B \le 0.5$ for which the network is unstable. We assume a packet is created in the end of a TS, and that packets are sent in the beginning of it.

Theorem 26: For C_A=1*pckt/TS*, the probability that buffer A is full at the beginning of a given *TS* (before services) is p_A.

Proof: Since packets from A have higher priority those packets don't wait in the buffer and every packet that arrives from queue A finds an empty buffer. Therefore, queue A gets service rate of C_A=1 packet per TS and $P_0^A=1-p_A$, $P_1^A=p_A$. Since link A sends one packet every TS, the state of queue A in the ith *TS* is the state of buffer A in *TS i+1*. Therefore the probability to find buffer A full is p_A. ■

Theorem 27: For C_A=1*pckt/TS* the probability that a packet arrives and finds buffer B full is modeled by

$$\frac{\left[p_A\left(1-P_0^B\right)\right]^2}{2-P_0^B\left(1+2p_A-p_AP_0^B\right)}$$

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*.

Let's explain the transitions: From *state 0*, if queue B is empty then we stay in this state, this happens with probability P_0^B. Otherwise, a packet will be sent and we move to *state 1*. From *state 1* we continue to *state 2* since the buffer is empty. From *state 2*, when queue B is empty no packet will depart from it and therefore we can either stay in *state 2* or go to *state 0*. If buffer A is empty, link C will serve buffer B and it will become empty, therefore we go to *state 0* with probability $q_A\cdot P_0^B$. On the other hand, if buffer A is full, then the packet in buffer B will need to wait and we stay in *state 2* with probability $p_A\cdot P_0^B$. When queue B is not empty a packet will depart from it and we move to *state 1* or *state 3*. If buffer A is empty, link C will serve buffer B and it will become empty, and therefore we go to *state 1* with probability $q_A\cdot(1-P_0^B)$. On the other hand, if buffer A is full, then the packet in buffer B will need to wait and we go to *state 3* with probability $p_A\cdot(1-P_0^B)$. Finally, when in *state 3*, since there is currently a packet on the way there will be no packet on the way in the next time slot. If buffer B gets service, then the packet on the way will enter the buffer, so it will remain full, and we go to *state 2*. If buffer B doesn't get service then the packet on the way will be thrown but obviously buffer B will remain full with the old packet and again we go to *state 2*. Therefore from *state 3* we only go to *state 2*.

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. We later justify this approximation which as demonstrated by the simulations, yields satisfactory results.

We now solve the Markov chain. We define π_i to be the probability of state $i \in \{0,1,2,3\}$. The steady state equations give:

$$(1-P_0^B)\cdot\pi_0 = q_A\cdot P_0^B\cdot\pi_2 \tag{31}$$

$$\pi_1 = (1-P_0^B)\cdot\pi_0+q_A\cdot(1-P_0^B)\cdot\pi_2 \tag{32}$$

$$\pi_3 = p_A\cdot(1-P_0^B)\cdot\pi_2 \tag{33}$$

In addition,

$$\pi_0+\pi_1+\pi_2+\pi_3 = 1 \tag{34}$$

Inserting Equations (31)-(33) in Equation (34) yields

$$\frac{q_A P_0^B}{\left(1-P_0^B\right)}\pi_2 + q_A\pi_2 + \pi_2 + p_A\left(1-P_0^B\right)\pi_2 = 1$$

$$\Rightarrow \pi_2\left(\frac{q_A P_0^B}{\left(1-P_0^B\right)} + q_A + 1 + p_A\left(1-P_0^B\right)\right) = 1$$

$$\Rightarrow \pi_2\left(\frac{q_A P_0^B + \left(2-p_A P_0^B\right)\left(1-P_0^B\right)}{\left(1-P_0^B\right)}\right) = 1$$

$$\Rightarrow \pi_2 = \frac{1-P_0^B}{2-P_0^B\left(1+2p_A-p_A P_0^B\right)}$$

Finally, in order for a packet to arrive to buffer B and to find a full buffer two things need to happen: first, the chain is in state 3, and at the same time, buffer A is not empty. The probability for that to happen is

$$p_A\pi_3 = p_A^2\left(1-P_0^B\right)\pi_2 = \frac{\left[p_A\left(1-P_0^B\right)\right]^2}{2-P_0^B\left(1+2p_A-p_A P_0^B\right)}. \qquad \blacksquare$$

Theorem 28: For C_A=1*pckt/TS* queue B is stable iff $p_B+p_A\pi_3<0.5$.

Proof: The probability for a packet to arrive and to find buffer B full is $p_A\pi_3$. 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 $p_B+p_A\pi_3$. Since the service rate of queue B is 1 packet per 2 timeslots, by definition queue B is stable iff $p_B+p_A\pi_3<0.5$.

Note that we could use an alternative proof, rather than increasing the arrival probability, we treat link B as if some of the work done by it is lost. The probability that work done by queue B is lost is $p_A\pi_3$, where each time work is lost queue B loses two *TS*s. Therefore the maximum effective utilization of link B is $1-2p_A\pi_3$ and for stability the arrival rate must be smaller than the service rate, namely $p_B<(1-2p_A\pi_3)/2$, i.e. $p_B+p_A\pi_3<0.5$. ■

Theorem 29: There exist arrival rates p_A, p_B for which the network with is stable C_A=0.5*pckt/TS* but unstable for C_A=1*pckt/TS*. Therefore, the CAP phenomenon also appears in store and forward networks, since one can add capacity and destabilize the network.

Proof: By example. Consider $p_A=p_B=0.48$. Theorem 25 shows that the network with C_A=0.5*pckt/TS* is stable for these arrival rates, and Theorem 28 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 $\pi_3<0.0417$. It can also be seen that π_3 is decreasing when P_0^B increases. Clearly, $P_0^B \le 1-2p_B=0.04$. Therefore $\pi_3\ge 0.11>0.04$ 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*. Define:

$$\tilde{p}_B \triangleq p_B + p_A\pi_3,$$
$$a \triangleq 4p_A, \qquad b \triangleq 2-4p_A\left(p_A+p_B\right),$$
$$c \triangleq 1-p_A-2p_B, \quad d \triangleq -p_B\left(1-p_A\right)$$
$$f\left(\tilde{p}_B\right) \triangleq a\cdot\tilde{p}_B^3 + b\cdot\tilde{p}_B^2 + c\cdot\tilde{p}_B + d$$

Then the network is stable iff $\tilde{p}_B < \frac{1}{2}$, and we show that $\tilde{p}_B$ is given by the solution of the equation $f\left(\tilde{p}_B\right)=0$ and thus *f* can tell when does the network is stable.

Theorem 30: When $p_B>0$, there is only one solution to the equation $f\left(\tilde{p}_B\right)=0$ in $\tilde{p}_B \in \left[p_B, 1\right]$, and the solution is in $\left[p_B, 0.5\right]$ iff $p_B + \frac{1}{2}p_A^2 < \frac{1}{2}$.

Proof: Obviously *f* is continuous and also,

$$\begin{aligned} f\left(\tilde{p}_B = p_B\right) &= 4p_A p_B^3 + 2p_B^2 - 4p_A p_B^2\left(p_A+p_B\right) \\ &\quad + p_B - p_A p_B - 2p_B^2 - p_B + p_A p_B \\ &= 4p_A p_B^2\left(p_B - p_A - p_B\right) = -4p_A^2 p_B^2 < 0 \end{aligned}.$$

Therefore to show there is only one solution we prove that $f\left(\tilde{p}_B\right)$ is increasing in $[p_B, 1]$ and that $f\left(\tilde{p}_B = 1\right) > 0$. First,

$$f\left(\tilde{p}_B = 1\right) = -\underbrace{4p_A^2}_{\le 1} + \underbrace{\left(3-3p_B\right)p_A}_{\ge 0.75} + \underbrace{\left(3-3p_B\right)}_{\ge 1.5} > 0$$

To prove *f* is increasing in $[p_B, 1]$ we show that in $\tilde{p}_B = p_B$ the first derivative of *f* is positive, and that in $[p_B, 1]$ the second derivative is also positive.

$$f'\left(\tilde{p}_B\right) = \frac{df}{d\tilde{p}_B} \triangleq 3a\cdot\tilde{p}_B^2 + 2b\cdot\tilde{p}_B + c$$

$$f'\left(p_B\right) = 12p_A p_B^2 + 2\left[2-4p_A\left(p_A+p_B\right)\right]p_B + 1 - p_A - 2p_B$$

$$= 4p_A p_B^2 + \left(2 - \underbrace{8p_A^2}_{\le 2}\right)p_B + \underbrace{\left(1-p_A\right)}_{>0} > 0$$

The second derivative of *f* is $f''\left(\tilde{p}_B\right) = \frac{d^2 f}{d\tilde{p}_B^2} = 6a\cdot\tilde{p}_B + 2b$, and since $6a>0$ it is a linearly increasing function. Therefore, to prove it is positive for any $\tilde{p}_B \in \left[p_B, 1\right]$ it is sufficient to show that it is positive when $\tilde{p}_B = p_B$.

$$f''\left(p_B\right) = 24p_A p_B + 2\left[2-4p_A\left(p_A+p_B\right)\right]$$
$$= 4 + 16p_A p_B - \underbrace{8p_A^2}_{\le 2} \ge 2$$

This proves there is only one solution to the equation $f\left(\tilde{p}_B\right)=0$ in $\tilde{p}_B \in \left[p_B, 1\right]$.

To complete the proof we show that $f\left(\tilde{p}_B = 0.5\right) > 0$ iff $p_B + \frac{1}{2}p_A^2 < \frac{1}{2}$.

$$\begin{aligned}
&0 < f(\tilde{p}_B = 0.5) = \\
&= \frac{1}{2}p_A + \frac{1}{2} - p_A(p_A + p_B) + \frac{1}{2} - \frac{1}{2}p_A - p_B - p_B(1 - p_A) \\
&= 1 - p_A^2 - 2p_B \\
&\Leftrightarrow p_B + \frac{1}{2}p_A^2 < \frac{1}{2}
\end{aligned}$$

By that the proof of the theorem is complete. ■

Theorem 31: When C_A=1*pckt/TS* queue B is stable iff $p_B + \frac{1}{2}p_A^2 < \frac{1}{2}$.

Proof: From Theorem 28 we know that the network is stable iff $\tilde{p}_B < \frac{1}{2}$. From queuing theory we know that for a stable network P_0^B=1-λ/μ, where λ is the average arrival rate and μ is the average service rate. In our case $\lambda = \tilde{p}_B$, μ=0.5 and therefore P_0^B=max{1-2 $\tilde{p}_B$, 0}. Let's assume the network is stable and place P_0^B in the definition of $\tilde{p}_B$:

$$\begin{aligned}
&\tilde{p}_B \triangleq p_B + p_A\pi_3 = p_B + \frac{p_A^2(2\tilde{p}_B)^2}{2-(1-2\tilde{p}_B)(1+2p_A - p_A(1-2\tilde{p}_B))} \\
&= p_B + \frac{4p_A^2\tilde{p}_B^2}{2-(1-2\tilde{p}_B)(1+p_A+2p_A\tilde{p}_B)} \\
&= p_B + \frac{4p_A^2\tilde{p}_B^2}{1-p_A+2\tilde{p}_B+4p_A\tilde{p}_B^2} \\
&\Leftrightarrow 0 = f(\tilde{p}_B) = 4p_A\tilde{p}_B^3 + \\
&\qquad [2-4p_A(p_A+p_B)]\tilde{p}_B^2 + \\
&\qquad (1-p_A-2p_B)\tilde{p}_B - \\
&\qquad p_B(1-p_A)
\end{aligned}$$

If there is a solution for the equation $f(\tilde{p}_B) = 0$ in the interval $\tilde{p}_B \in [p_B, 0.5]$ then our assumption was correct and the network is stable, otherwise our assumption was wrong and the network is not stable. In Theorem 30 we saw that such solution exists iff $p_B + \frac{1}{2}p_A^2 < \frac{1}{2}$ and therefore the proof is complete. ■

The last result is actually intuitive. Note that 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 π_3=0.5·p_A. Therefore the arrival rate to queue B is $p_B + \frac{1}{2}p_A^2$, and if it is smaller than 0.5 then indeed queue B is unstable and P_0^B=0.

As said, it is not accurate to use P_0^B in the Markov chain and in some cases we should have used the probability that queue B is empty given that there was a departure two or more TSs ago. However, since the last probability is not known to us we take P_0^B as an approximation. A better approximation can be to take the probability that queue B is empty when

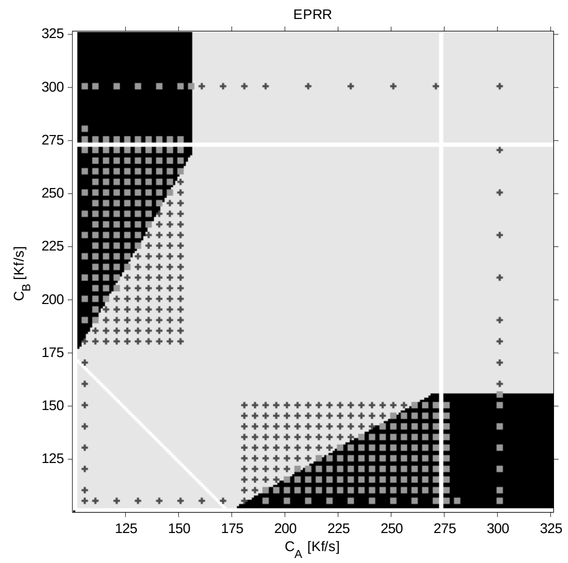

Figure 4: Wormhole results for R_A=R_B=100*Kflit/sec*. Unstable (black) and stable (gray) configurations according to analysis are compared to unstable (squares) and stable (crosses) configurations according to simulations. The arbitration used is EPRR. The numbers represent the different cases.

there was a departure two TSs ago, namely $\tilde{P}_1^B(1-p_B)^2$, where $\tilde{P}_1^B$ is the probability that there was exactly one packet in the queue given that it is not empty (because a packet just left). However, this better approximation will not be much more accurate since

$$\begin{aligned}
&\tilde{P}_1^B = P(Q_B = 1 \mid Q_B \ge 1) = \\
&= \frac{P(Q_B \ge 1 \mid Q_B = 1)P(Q_B = 1)}{P(Q_B \ge 1)} = \frac{P_1^B}{1-P_0^B}.
\end{aligned}$$

and since we are concerned in the case where the load is high then P_0^B ,P_1^B <<1 and $P_0^B \approx P_1^B$ (if we consider an M/M/1 model for example), therefore $\tilde{P}_1^B(1-p_B)^2 \approx P_1^B(1-p_B)^2 < P_0^B$. If we take a look at the Markov chain we see that taking higher value for P_0^B should help the stability of the chain. And therefore we actually found sufficient conditions for the instability of the network. However as simulations suggest the analysis is accurate and the conditions are also necessary.

V. Simulations Results

We now present simulations results that demonstrate the existence of the CAP phenomenon and confirm the foregoing analysis.

A. Wormhole Networks

An OPNET-based wormhole simulator [19][21][25] 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 queue length on each interval. In a stable network, the average queue length is bounded, where in an unstable network the average queue length typically increases near-linearly.

1) Simulations for example 1 – EPRR only

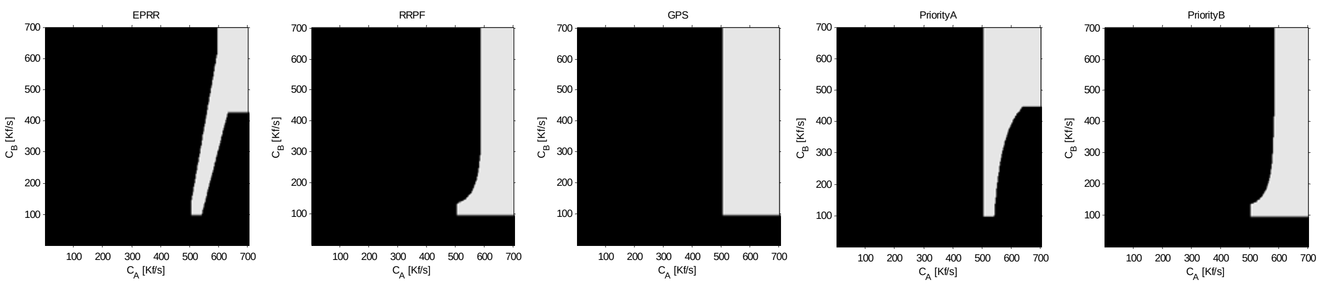

Figure 5: Unstable (black) and stable (gray) configurations according to wormhole analysis when R_A=500*Kflit/sec*, and R_B=100*Kflit/sec*. The figures refer (from left to right) to EPRR, RRPF, GPS, Priority (A has higher priority), Priority (B has higher priority)

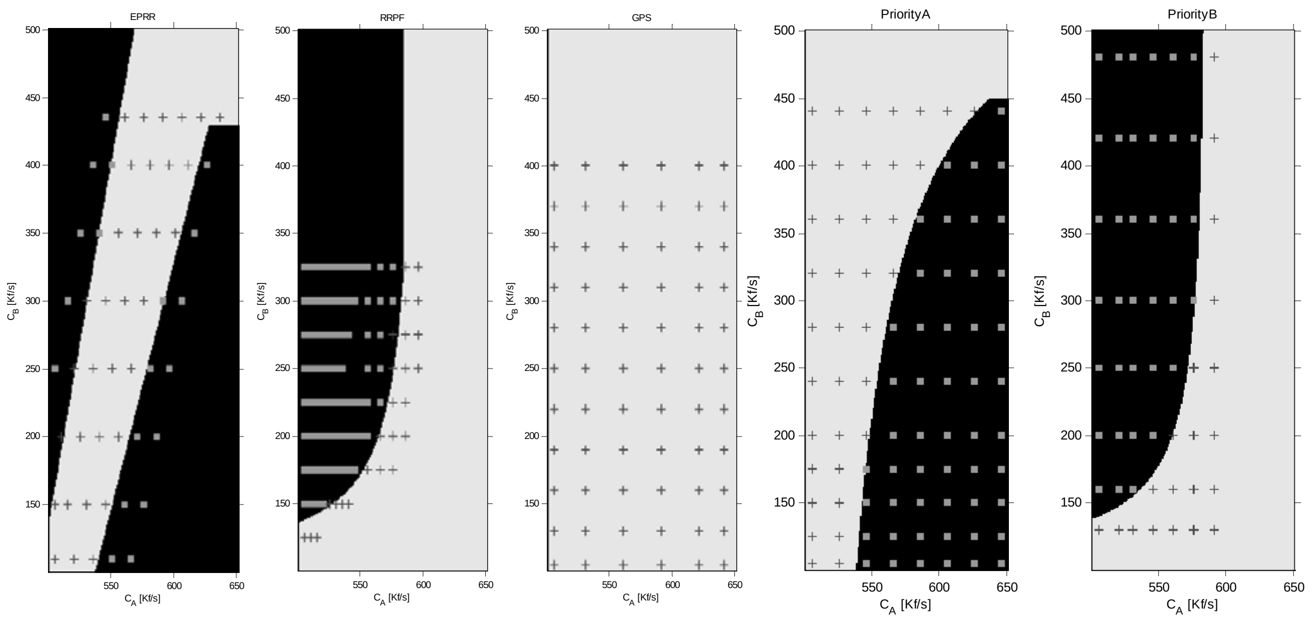

Figure 6: Wormhole results for R_A=500*Kflit/sec*, R_B=100*Kflit/sec*. Unstable (black) and stable (gray) configurations according to analysis are compared to unstable (squares) and stable (crosses) configurations according to simulations. The figures refer (from left to right) to EPRR, RRPF, GPS, Priority (A has higher priority), Priority (B has higher priority)

First we show the existence of the phenomenon in a symmetric network which uses the EPRR arbitration. We use the same parameters as in *Example 1*, namely L=1000 *flit/pckt*, N=10 *bit/flit*, r_A=r_B=100 *pckt/sec*, implying R_A=R_B=100 *Kflit/sec*. In addition B_i=16 *flit* and C_R=272 *Kflit/sec*. Note that for those parameters both RRPF and GPS yield a stable network.

Figure 4 shows simulation results on areas where the necessary stability conditions are fulfilled. The numbers in this figure represent the case that fits the configuration as defined in section III.C; the white lines are used as delimiters between the cases. All links capacities and rates are in *Kflit/sec*.

Let's take a look in Figure 4 at the lowest horizontal line of simulations, where C_B=105. On this line we start with C_A=105, and change it until it reaches 300. Therefore we start with case (1) where $C_A+C_B \le C_R$. When C_A satisfies the equation: $C_A+C_B=C_R$, namely C_A=167 we move to case (3). It can be seen that for case (1) the network is stable and that at some point in case (3) the network becomes unstable. When C_A satisfies the equation $C_A=C_R$, namely C_A=272 we move to case (2) where the network remains unstable.

In order to stabilize the network without changing capacity we can use RRPF arbitration or increase the size of the buffers so they will never get full.

We may also stabilize the network by changing capacity. Obviously each suggested solution must also satisfy the necessary conditions. One option is to assign equal capacities to all links. This will lead us to case (4) which is always stable. Another option is to decrease C_A below ~180, which leads to stable region in case (3). We can also increase only the capacity C_R to 405 and move to case (1), or increase both C_A and C_R to 1590 and stay in case (2), but note that link R should only transmit 200.

The last option would be to increase C_B to capacity greater than ~155 and also remain in case (2), this option is represented in Figure 4 by the vertical line of simulations at the right. We found earlier that for this configuration when we are in case (2) the demand for stability is C_B>155 and as can be seen in the picture when C_B>155 the network is still not stable where for C_B=160 it is stable.

2) *Simulations of all arbitrations*

Now we consider asymmetric traffic requirements and simulate all the arbitrations we proposed. We use the following parameters, 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_R=636 *Kflit/sec*. Since $R_B=5R_A$, when using the GPS arbitration and both buffers are not empty the router transmit 5 flits from A for

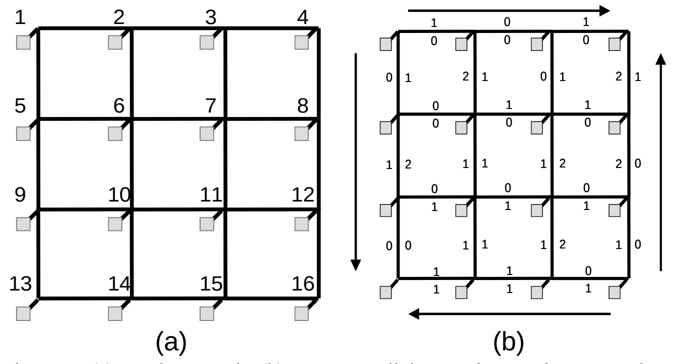

Figure 7: (a) Mesh network. (b) Necessary link capacity requirements when using XY routing.

TABLE 2 TRAFFIC MATRIX FOR MESH NETWORK

Source	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Destination	6	14	12	16	9	8	15	4	5	1	2	11	10	3	13	7

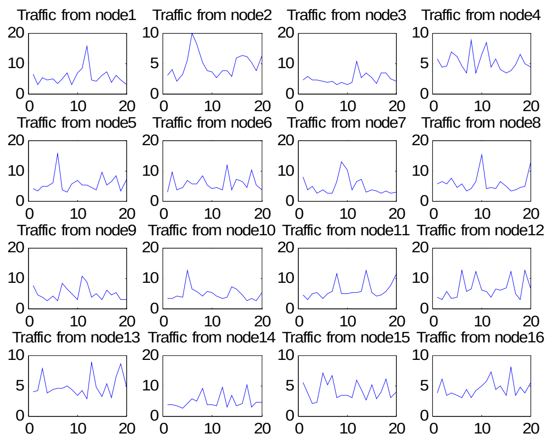

Figure 8: Mesh simulation results before increasing the capacity (average queue size as a function of the interval)

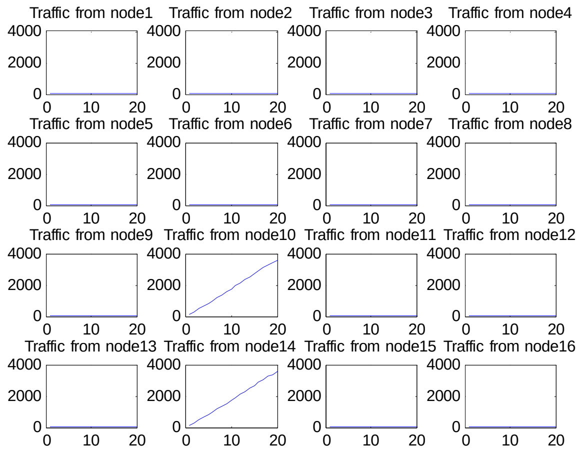

Figure 9: Mesh simulation results after increasing the capacity (average queue size as a function of the interval)

every 1 packet it transmits from B.

Figure 5 shows wormhole configurations in which the mathematical analysis yields a stable or unstable network (colored gray or black, respectively). For priority we used the approximated model (Theorem 14 and Theorem 17) and not only the sufficient conditions (Theorem 13 and Theorem 16). Figure 6 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 $C_A \geq C_R$=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. 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.

3) Simulations for a mesh network – EPRR only

In this example we consider a more complex network with the EPRR arbitration and show the existence of the phenomenon is not limited to the simple network we presented.

Figure 7(a) shows the 4x4 mesh network we are dealing with. Suppose the network uses XY routing and that we number the links and the nodes as shown in Figure 7(a). We define one traffic unit with the following parameters: L=500 *flit/pckt*, r=100 *pckt/sec*, thus R=50 *Kflit/sec*, and assume that the traffic matrix is a permutation as shown in Table 2. Finally, assume that in each link a VC is always available, namely the number of VC's on each link is equal to the number of flows going through the link.

Figure 7(b) shows the capacity demands for the network (in traffic units). On each vertical link the number on the right (respectively left) specifies the amount of bandwidth that should travel on it from bottom up (up to bottom). In the same manner on each horizontal link the number above it (respectively below it) specifies the amount of bandwidth that should travel on it from left to right (right to left).

We simulated the network for 100 seconds which were divided to 20 intervals.

Figure 8 shows the average queue length as a function of the interval for each one of the nodes when the capacity allocation for each link is 1.1 of the bandwidth that needs to cross it. As can be seen, the average queue size stays bounded which means the network is stable.

Now we decide that traffic from nodes 16 and 9 doesn't arrive fast enough to nodes 7 and 5 respectively. Therefore we increase the capacity of the links along the path of those flows to 2.2 (which was the maximum capacity given to a link earlier).

Figure 9 shows simulation results for the last capacity allocation and as can be seen the network is now unstable. The queue lengths of nodes 9 and 16 are very small now. However,

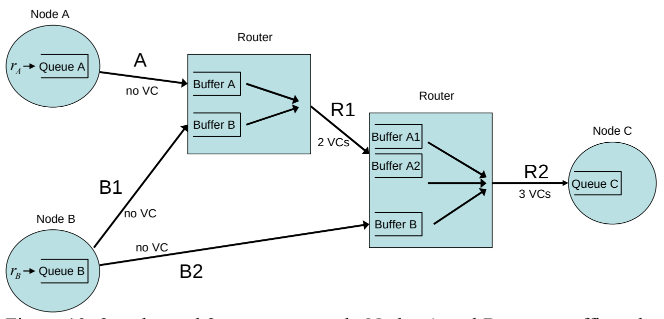

Figure 10: 3 nodes and 2 routers network. Nodes A and B create traffic to be sent to node C in rates 1 and 2 respectively.

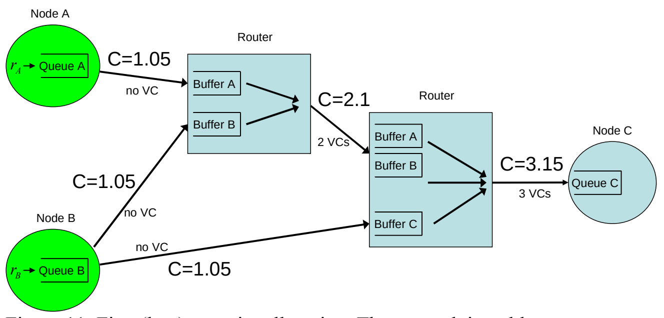

Figure 11: First (low) capacity allocation. The network is stable.

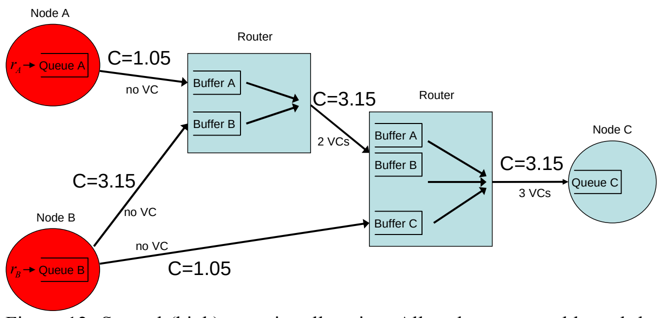

Figure 12: Second (high) capacity allocation. All nodes are unstable and the network is unstable.

queue 10 is not stable because the utilization of the link between router 10 and router 9 may not be 100% due to traffic that travels from node 9 to node 5 and therefore this link cannot transmit all the traffic from node 10. The problem is exactly as the one we presented in case (2) of EPRR (Section III.C), only that link R is replaced by the link between routers 9 and 5, link B is replaced by the link between routers 9 and 10, and finally link A is replaced by the link between node 9 to router 9. The same problem also occurs for node 14, where the utilization of the link between router 14 and router 15 may not be 100% because of the traffic that travels from node 16 to node 7.

4) *All nodes can suffer from increased capacity- EPRR*

In our last wormhole example, we show how all nodes in a network can suffer from an increased capacity.

Figure 10 presents a network with 3 nodes and 2 routers and with the EPRR arbitration. For the links, we use the notations as they appear on Figure 10. Assume packet size is much longer than buffer size and that nodes A and B create packets with respective rates 1 and 2, all to be sent to node C. All flits from A traverse the same route (Node A, Router 1, Router 2, Node C), whereas every even flit created in node B is routed through link B1, and every odd packet is routed through link B2. Therefore node B creates packets with rate 1 to either one of the links B1 and B2. Finally assume queue B has two reading ports so it can send two packets to Node C at the same time in two different paths, and that a VC is always available.

Figure 11 shows the low-capacity allocation we start with. For this capacity allocation we give each link 1.05 times its bandwidth requirements. It is easy to see that for this capacity allocation the network is stable (similar to case (1) in III.C).

Figure 12 shows the high capacity allocation where we add capacity to links B1 and R1 and allocate a capacity of 3.15 (instead of 1.05 and 2.1 respectively). Due to the increased capacity, the path (B1, R1, R2) is busy for approximately a third of the time transmitting traffic (even numbered packets) from node B. Therefore, as in case (2) in III.C, links A and B2 may not work for about a third of the time and they cannot transmit all the traffic presented to them.

B. *Store and Forward Networks*

Figure 13 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 31. It can be seen that the analysis is 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 14 shows four additional simulations, performed using OPNET. The following arbitrations have been used in the router:

Priority: The router gives packets from A higher priority. Thus, the router always tries to serve buffer A and only is it is empty it serves buffer B.

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 3 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.

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 link capacity until the stable point is found.

Further, we proved that in some stable networks, any small deviation from an optimal link capacity, either 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.

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.

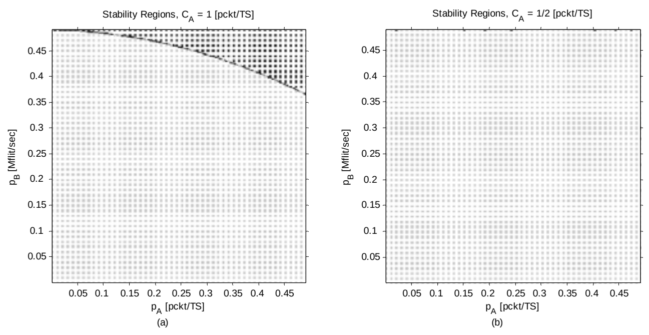

Figure 13: Store and Forward simulation results. Each square represent a simulation. Gray (Black) squares represent stable (unstable) simulations. (a) C_A=1, analysis suggests that every configuration above the black line is unstable. (b) C_A=0.5, analysis suggests every configuration is stable

TABLE 3 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	Exhaustive	RR	Exhaustive

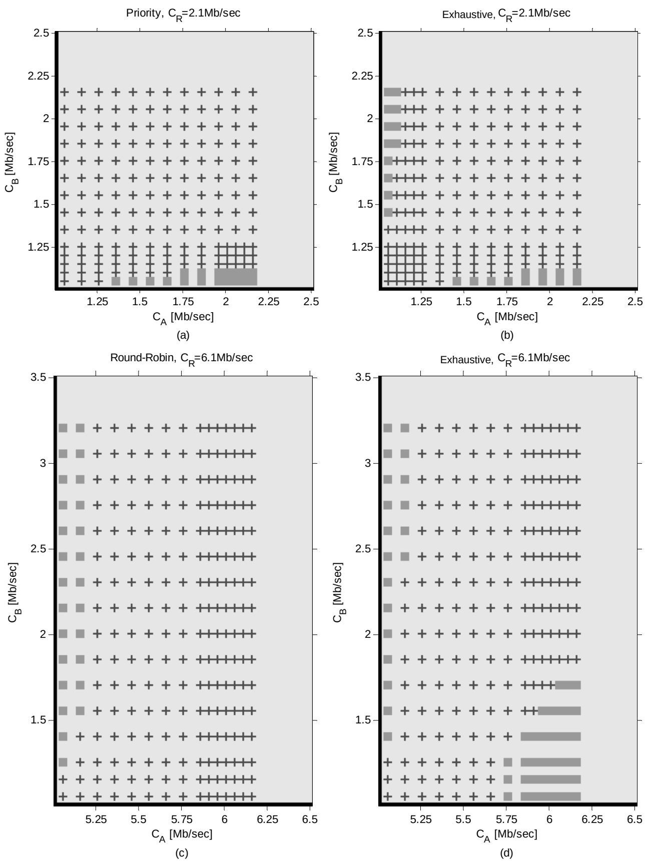

Figure 14: Simulations results for Store and Forward. Parameters can be seen in Table 3. Crosses (Squares) represent stable (unstable) configurations according to simulations. Adding capacity may destabilize the network.

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

[19] A. Baron, R. Ginosar, I. Keslassy, I. Walter, 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] 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.

[22] 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'l Conf. Comm. in Computing*, June 2000.

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

[24] 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.

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

[26] 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

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