Capacity of Byzantine Agreement: Tight Bound for the Four Node Network [*]

Guanfeng Liang and Nitin Vaidya
Department of Electrical and Computer Engineering, and
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
gliang2@illinois.edu, nhv@illinois.edu

Technical Report

February 1, 2010

1 Introduction

In this report, we consider the problem of maximizing the throughput of Byzantine agreement. Byzantine agreement is a classical problem in distributed computing, with initial solutions presented in the seminal work of Pease, Shostak and Lamport [1, 2]. There has also been more recent work on designing multicast algorithms that can survive Byzantine attacks (e.g., [3]).

Many variations on the Byzantine *agreement* problem have been introduced in the past, with some of the variations also called *consensus*. We will use the original definition of Byzantine agreement in our discussion. In particular, we consider a network with one node designated as the *sender* or *source*, and other nodes designated as the *peers*. The goal here is for all the fault-free nodes to "agree on" the value being sent by the sender. In particular, the following conditions must be satisfied:

- *Agreement:* All fault-free peers must agree on an identical value.

[*]This research was supported in part by Army Research Office grant W-911-NF-0710287

- *Validity:* If the sender is fault-free, then the value greed on by the peers must be identical to sender's value.

Once a node agrees on a certain value, it cannot change its decision. It is customary to also include a *termination* condition [4], which states that the peers must eventually agree on a value. We will revisit the notion of termination below.

Our goal in this work is to design algorithms that can improve *throughput* of agreement. The notion of throughput here is similar to that used in the networking/communications literature on unicast or multicast traffic. We now define *agreement throughput* (or throughput of agreement) somewhat more formally.

When defining throughput, the "value" referred above in the definition of agreement is viewed as an infinite sequence of bits. At each peer, we view the agreed information as being represented in an array of infinite length. Initially, none of the bits in this array at a peer have been agreed upon. As time progresses, the array is filled in with agreed bits. In principle, the array may not necessarily be filled sequentially. For instance, a peer may agree on bit number 3 before it is able to agree on bit number 2. Once a peer agrees on bit number i in the array, that agreed bit cannot be changed. In the discussion below, we assume that agreement algorithm begins execution at time 0. The system is assumed to be synchronous.

Throughput: For defining throughput, we consider the largest *prefix* of the array at all the peers that has been agreed upon. In particular, suppose that at time t, all the peers have agreed upon bits 0 through $b(t)$, and at least one peer has not yet agreed on bit number $b(t)-1$, then the largest agreed prefix at time t has size $b(t)$. Agreement throughput T is defined as

$$T = \lim_{t\to\infty} \frac{b(t)}{t} \tag{1}$$

Although we assume that the information granularity is a bit, the definition can be modified trivially for symbols of larger size.

We do not explicitly define a termination condition. However, achieving non-zero throughput implicitly requires that agreement be eventually achieved on an infinite prefix of the information. In the above definition, we also do not restrict the delay in reaching the agreement. However, the definition may be modified to include additional constraints.

Capacity: Capacity of agreement in a given network, for a given sender and a given set of peers, is defined as the supremum of all achievable throughputs of agreement between the given sender and the peers.

Note that, in general, not all nodes in the network may be peers. That is, the network may include nodes that are neither sender nor peers. Such nodes may assist in achieving agreement.

The above definitions of throughput and capacity can be extended to other forms of agreement or consensus as well. For instance, *consensus* is often defined as agreeing on an identical value as a function of the values being proposed by all the peers (so, all the peers act as "senders"). In particular, if all the fault-free peers propose an identical value, then the agreed value should be this particular value. It should be easy to see that the above definition of throughput can be extended to this scenario as well. It should also be easy to see that the actual throughput for the different versions of consensus or agreement may be quite different.

The above definitions can be extended to the more general problem of *function computation*, where the agreed value may be an arbitrary function of the values proposed by the various nodes, and also to the notion of *approximate agreement* (in case of approximate agreement, one can trade-off throughput with "accuracy" of agreement).

While all of these issues are quite interesting, in this report, we consider the simplest instance of the agreement problem: a fully connected four node network.

2 Necessary Conditions

In our previous report [5], we introduced the following two necessary conditions for an agreement throughput of R bits/unit time to be achievable:

- If one of the peers is removed from the network, then the min-cut from the source to any remaining peer must be at least R.
- The max-flow to one of the peers from the other peers, with the source removed, must be at least R.

The reason for these conditions to be necessary is discussed in detail in our previous report [5].

3 Tight Lower Bound for the Four Node Network

In this report, we provide tight lower bound of the agreement capacity of the four node network.

In particular, consider a synchronous system consisting of four nodes, namely, S, A, B and C, with node S acting as the sender, and nodes A, B and C being the peers. At most one of these four nodes may be faulty.

We assume that each pair of nodes is connected by a pair of directed point-to-point links. Each point-to-point link has a certain *link capacity*, defined as the maximum rate at which information may be transmitted over that link.[1]

Figure 1 shows the four-node network. The labels near the various links denote the link capacities.

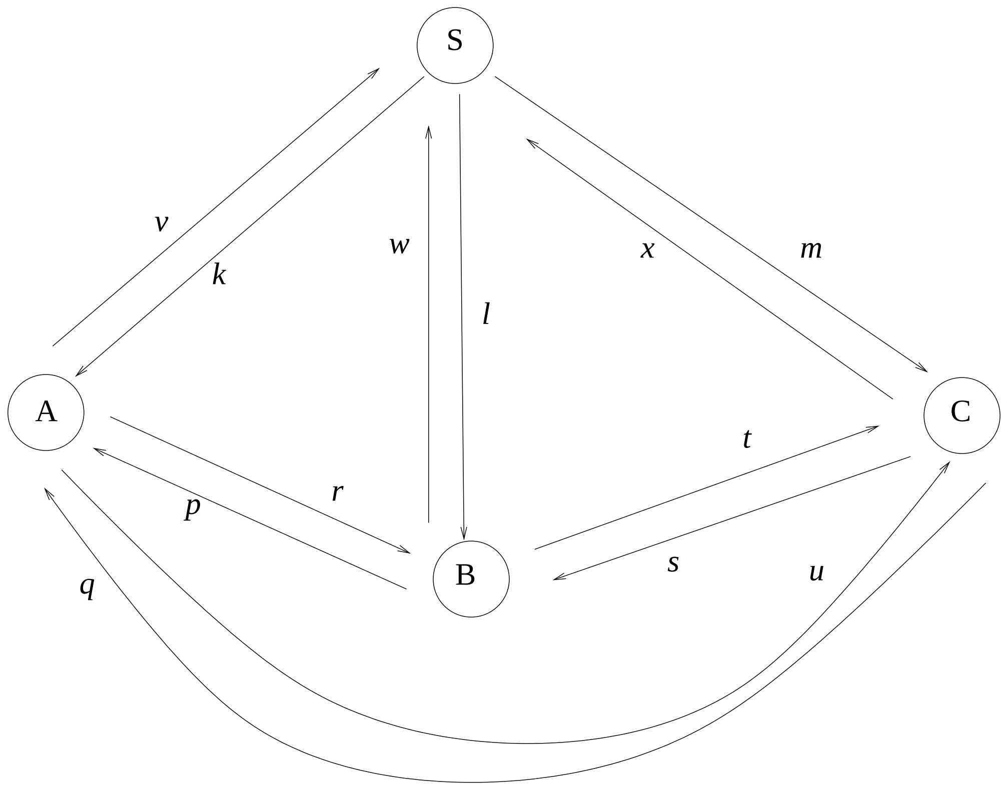

Figure 1: Four node network: Labels denote the link capacities.

[1]In some instances, it is not necessary to have links in both directions between certain pairs of nodes. For brevity, we ignore this possibility here, and consider the case where links exist in both directions. In fact, we assume that a non-zero link capacity is available for all directed links.

4 Byzantine Agreement Algorithm for the Four Node Network

4.1 Sufficient Link Capacity Constraints

We will present a Byzantine algorithm for the network in Figure 1. The algorithm can achieve agreement throughput arbitrarily close to R bits/unit time, provided that the following sets of conditions are true.

- If one of the peers is removed from the network, the min-cut from the source to any remaining peer must be greater than R. This implies

$$k + l > R \tag{2}$$
$$l + m > R \tag{3}$$
$$m + k > R \tag{4}$$
$$p + k > R \tag{5}$$
$$q + k > R \tag{6}$$
$$r + l > R \tag{7}$$
$$s + l > R \tag{8}$$
$$t + m > R \tag{9}$$
$$u + m > R \tag{10}$$

- The max-flow to one of the peers from the other peers, with the source removed, must be at greater than R. This implies

$$p + q > R \tag{11}$$
$$r + s > R \tag{12}$$
$$t + u > R \tag{13}$$

  The above conditions are *necessary* for throughput of R to be achieved.

- We also impose the constraint that the capacity of each directed link is more than 0. This allows each node to transmit a fixed number of bits to another node within a finite interval of time.

The agreement algorithm has four different modes of operation. We will number these modes as I, II, III, IV. As seen later, repeated (and pipelined) execution of our algorithm below can be used to achieve the desired throughput. The network starts in mode I, and may change modes over time as the algorithm is executed multiple times. The mode number does not decrease over time[2].

- Mode I: *Fault-free*: The network starts operation in this mode.
- Mode II: Failure known to be within a set containing two peers.
- Mode III: Failure known to be within a set containing a peer and the source.
- Mode IV: Failure identified and known to all good nodes

The 1 unit of data is divided into 6 parts or "packets", named D through I. For the packet sizes below to be meaningful, we assume that $k, l, m \leq 1$.

- D: S sends to every peer, of size d
- E: S sends to A and B, of size e
- F: S sends to B and C, of size f
- G: S sends to A and C, of size g
- H: S sends to A, of size h
- I: S sends to B, of size h
- $H \oplus I$: S sends to C, of size h

The packet sizes are normalized by R. Thus, normalized packet size $(l-k)$ really means $(l-k)R$ bits. We assume that with appropriate scaling factors, the normalized packet sizes can all be turned into integers (this is true if k, l, m are rational numbers). Note that the normalized packet sizes for packets D, E, F, G, H, I must add up to 1, i.e.

$$d + e + f + g + 2h = 1 \tag{14}$$

We also assume that k, l, m are all saturated, i.e.,

$$d + e + g + h = k \tag{15}$$
$$d + e + f + h = l \tag{16}$$
$$d + f + g + h = m. \tag{17}$$

[2]The algorithm can be modified somewhat to allow for the possibility of node repair. In this case, the mode number may indeed decrease. In this report, we do not consider the possibility of repair, and assume that at most one node can fail over the infinite time duration.

It is possible that a downlink capacity from S is so large such that it will not be saturated. In this case, we under utilize that link and have the effective capacity being used satisfy the above equations.

In general, our algorithm executes as follows: the network starts in mode I (fault free), and the information from S is scheduled as we are going to describe in the next section. The schedule is designed such that any misbehavior of a single node (source or peer) will be detected by at least one peer. Once any node detects an attack, the system enters a broadcast phase, in which every node broadcast the information it has sent and received to every other node, using the traditional Byzantine agreement algorithm. After the broadcast phase, the location of the faulty node will be narrowed down to within a set of two peers (mode II), or a set of a peer and the source (mode III), or a single node (mode IV). If the network enters mode II or mode III, a modification of the schedule in mode I is applied. This modified schedule also ensures any misbehavior be detected. Once any node detects an attack in mode II or mode III, the system enters the broadcast phase for the second time. After the second broadcast phase, the location of the faulty node is identified and known to every good node (mode IV). Once the network enters mode IV, the faulty-free peers either agree on some default value (source known to be faulty), or ignore the information from the faulty peer and agree with the value transmitted by source.

4.2 Operation in Mode I

The operation proceeds in three rounds. And the operation differs slightly depending on the value of $k + l + m$.

4.2.1 When $k + l + m \leq 2$

Set the packet sizes as follows:

$$\begin{aligned} d &= 0 && (18)\\ e &= k + l - 1 && (19)\\ f &= l + m - 1 && (20)\\ g &= k + m - 1 && (21)\\ h &= 2 - (k + l + m). && (22) \end{aligned}$$

According to the necessary conditions stated previously (Equation 2 to Equation 4), all the packet sizes are non-negative. Since $d = 0$, there is in fact no packet D.

Round 1: Sender S transmits as follows:

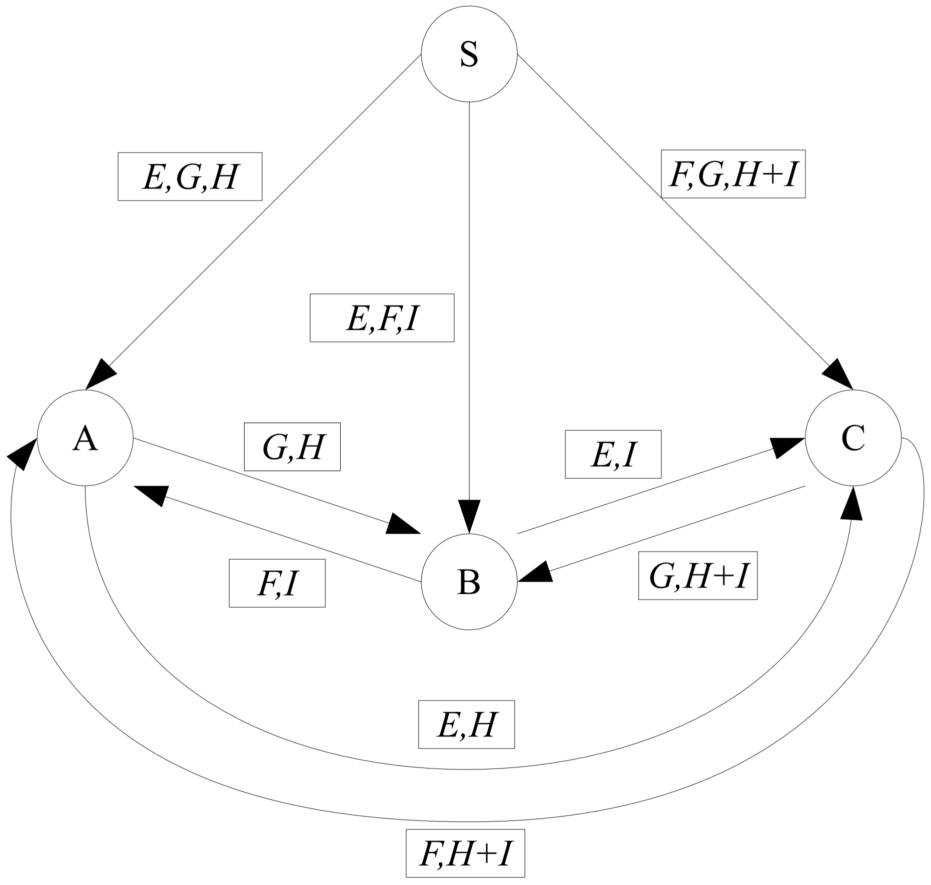

Figure 2: Four node network when $k + l + m/le2$

E, G, H to node A,

E, F, I to node B, and

$F, G, H \oplus I$ to node C.

The number of bits sent on links from S to the peers are as follows:

link SA$= e + g + h = k$, link SB$= e + f + h = l$, and link SC$= f + g + h = m$.

Round 2: The peers send information to each other according to the following schedule. The schedule is illustrated in Figure 2

- Links AB and BA: Schedule packets G, H on link AB, and packets F, I on link BA.

Size of G, H together is $1-l$, and size of F, I is $1-k$. Thus we are within the capacity of r and p of links AB and BA, respectively.

- Links BC and CB: Schedule packets E, I on link BC, and packets $G, H \oplus I$ on link CB. Size of E, I together is $1-m$, and size of $G, H \oplus I$ is $1-l$. Thus we are within the capacity of t and s of links BC and CB, respectively.
- Links AC and CA: Schedule packets E, H on link AC, and packets $F, H \oplus I$ on link CA. Size of E, H together is $1-m$, and size of $F, H \oplus I$ is $1-k$. Thus we are within the capacity of u and q of links AC and CA, respectively.

Round 3: Assessment The peers use the information they have received from S and other peers to determine whether there is a "mismatch" (or inconsistency) between their information and the information of the peers. It can be shown that given the schedules described in previous sections, any misbehavior by a single node will result in at least one peer node sees a "mismatch". We now explain how each packet can be checked.

- Packet E: C receives two complete copies of packet E from A and C. If either A or B is faulty and sends a "bad" E, C will detect it. If S is faulty and sends different E's to A and B, C will detect it.
- Packets F, G: Similar to packet E, any misbehavior on packet F will be detected by node A; and any misbehavior on packet G will be detected by node B.
- Packets H, I: Every peer receives $H, I, H \oplus I$, which form a (3,2) error detection code, from S and the other two peers. If A is faulty and sends a "bad" H to B, B will detect it; if A sends a "bad" H to C, C will detect it. Similarly, if B or C is faulty and corrupts I or $H \oplus I$, it will be detected by at least on of the other two peers. If S is faulty and sends out these packets such that $H_A + I_B \neq (H \oplus I)_C$ (here we use the subscript to distinguish the packets S sends to each peer), every peer will detect it.

4.2.2 Round 1 and Round 2, When $k + l + m > 2$

Set the packet sizes as follows:

$$\begin{aligned} d &= k+l+m-2 && (23)\\ e &= 1-m && (24)\\ f &= 1-k && (25)\\ g &= 1-l && (26)\\ h &= 0. && (27) \end{aligned}$$

According to the necessary conditions stated previously, all the packet sizes are non-negative. Since $h = 0$, there is in fact no packets $H, I, H \oplus I$.

Round 1: Sender S transmits as follows:

D, E, G to node A,

D, E, F to node B, and

D, F, G to node C.

The number of bits sent on links from S to the peers are as follows:

link SA= $d + e + g = k$, link SB= $d + e + f = l$, and link SC= $d + f + g = m$.

Round 2: The peers send information to each other according to the following schedules.

- Links AB and BA: Schedule packets G on link AB, and packets F on link BA. Size of G is $1 - l$, and size of F is $1 - k$. Thus we are within the capacity of r and p of links AB and BA, respectively.
- Link BC and CB: Schedule packets E on link BC, and packets G on link CB. Size of E is $1 - m$, and size of G is $1 - l$. Thus we are within the capacity of t and s of links BC and CB, respectively.
- Link AC and CA: Schedule packets E on link AC, and packets F on link CA. Size of E is $1 - m$, and size of F is $1 - k$. Thus we are within the capacity of u and q of links AC and CA, respectively.

So far, we have not scheduled packet D yet. The schedule for packet D depends on the distribution of the capacity of the links between peers:

- If $p + q + r + u \geq 2 - k$, $p + r + s + t \geq 2 - l$, and $q + u + s + t \geq 2 - m$:

  Let z_1, z_2, z_3 be the remaining capacity on link pairs (AB,BA), (BC,CB), and (AC,CA), respectively, i.e.

$$z_1 = p + r - (1 - k) - (1 - l) \tag{28}$$

$$z_2 = s + t - (1 - l) - (1 - m) \tag{29}$$

$$z_3 = q + u - (1 - k) - (1 - m). \tag{30}$$

Given $p+q+r+u \geq 2-k$, we have

$$\begin{aligned} z_1+z_3 &= p+r-(1-k)-(1-l)+q+u-(1-k)-(1-m) && (31)\\ &= p+q+u+r-4+2k+l+m && (32)\\ &\geq k+l+m-2=d. && (33)\end{aligned}$$

Similarly, we also have $z_1+z_2 \geq d$ and $z_2+z_3 \geq d$. Moreover, from $p+q+r+u+s+t \geq 3$, we have

$$\begin{aligned} z_1+z_2+z_3 &= p+q+r+u+s+t-6+2(k+l+m) && (34)\\ &\geq 2(k+l+m)-3=2d+1. && (35)\end{aligned}$$

Without loss of generality, assume that $z_1 \leq z_2 \leq z_3$.

If $d \leq z_2$, there is enough capacity left for packet D between B and C as well as between A and C. Schedule part of D on BC, and remaining part of D on CB, such that all of D is scheduled, and the capacity of links BC and CB is not exceeded. The similar exchange of D is performed between A and C.

If $d > z_2$, then $d > z_1$, and $z_3 \geq 2d+1-z_1-z_2 > 1$. There is capacity left on links AC and CA together to transmit packet D. Schedule part of D on AC, and remaining part of D on CA, such that all of D is scheduled, and the capacity of links AC and CA is not exceeded. Also, since $z_1+z_2 \geq d$, there is capacity left on links AB, BA, BC, and CB together to transmit packet D. Schedule parts of D on AB, BA, BC, and CB such that all of D is scheduled, and the capacity of links AB, BA, BC, and CB is not exceeded.

- If $p+q+r+u < 2-k$:

  Let $p+q+r+u = 2-k-\delta$, for some $\delta > 0$. From $p+q+r+s+t+u \geq 3$, we have

$$z_2 \geq 3-(2-k-\delta)-(1-l)-(1-m) = k+l+m-1+\delta > 1+\delta. \qquad (36)$$

  So there is capacity left on links BC and CB together to transmit packet D. Schedule part of D on BC, and remaining part of D on CB, such that all of D is scheduled, and the capacity of links BC and CB is not exceeded. Notice that since $d \leq 1$, there is at least δ capacity still left on links BC and CB together.

  Meanwhile, the remaining capacity on links AB, BA, AC, and CA is

$$2-k-\delta-2(1-k)-(1-l)-(1-m) = k+l+m-2-\delta = d-\delta. \qquad (37)$$

  So we can schedule $d-\delta$ bits of D on links AB, BA, AC, and CA together such that the capacity of these links is not exceeded.

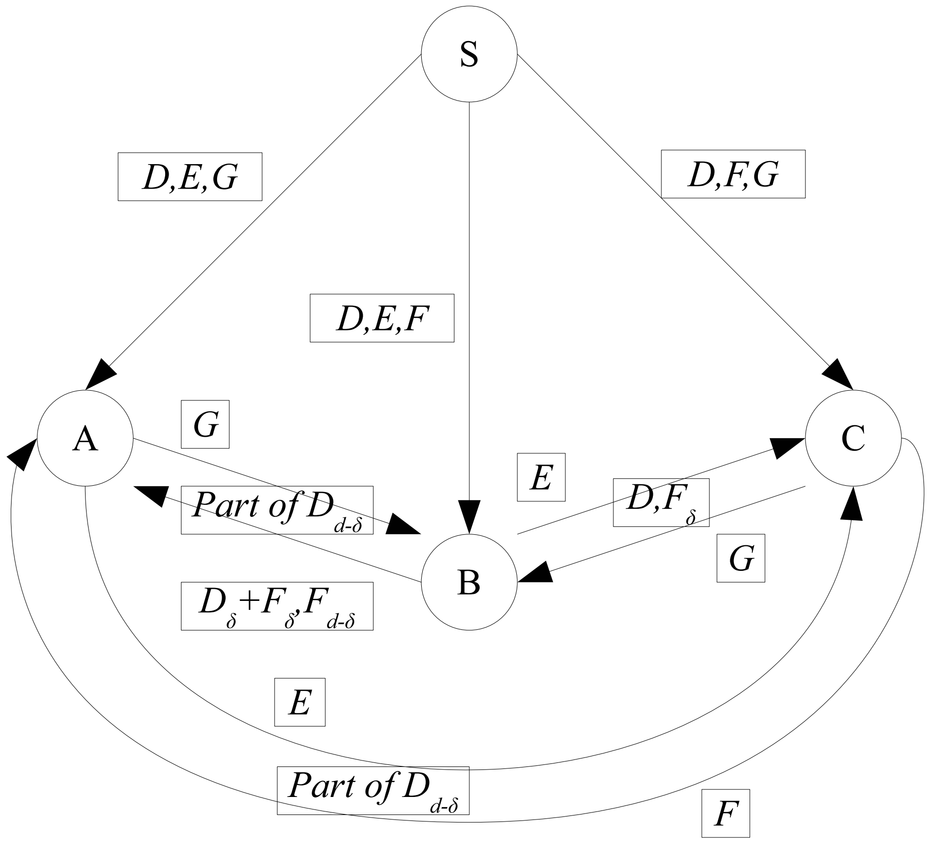

Figure 3: Four node network when $k + l + m > 2$ and $p + q + r + u < 2 - k$

Notice that on links BA and CA, 2 copies of packet F have been transmitted, by B and C. Also, the total capacity of BA and CA must be at least 1; then there is at least $1 - 2(1 - k) = 2k - 1$ capacity to transmit bits of D to A. Hence $d - \delta \geq 2k - 1$, and

$$\begin{aligned}
\delta &\leq d - (2k - 1) && (38)\\
&= k + l + m - (2k - 1) && (39)\\
&= l + m - k - 1 && (40)\\
&\leq 1 - k. && (41)
\end{aligned}$$

Equation 41 follows from Equation 40 because $l + m \leq 2$. This means the remaining segment of D is smaller than F. Denote the remaining δ bits of D (not scheduled on AB, BA, AC, CA) as D_δ, the first δ bits of F as F_δ, and the remainder of F as $F_{d-\delta}$. Here we need to modify the schedule on link BA in Round 2 a little bit. Instead of packet F, schedule $D_\delta \oplus F_\delta$ and $F_{d-\delta}$ on link BA in Round 2. Moreover, remember that there is at least δ capacity left on BC and CB together, so B and C will also exchange F_δ over links BC and CB.

- If $p+r+s+t < 2-l$ or $q+u+s+t < 2-m$, similar to the case when $p+q+r+u < 2-k$.

Round 3: Assessment

- If $p+q+r+u \geq 2-k$, $p+r+s+t \geq 2-l$, and $q+u+s+t \geq 2-m$
  - Packets E, F, G: they are checked in the same way as in the case when $k+l+m \leq 2$.
  - Packet D:

    If $d \leq z_2$, D is fully exchanged between node B and C, as well as between A and C. If A is faulty and sends some "bad" bits of D to C, C will detect it. Similarly, B and C will be detected by at least on peer if they are faulty and sends some bad bits of D in round 2. If S is faulty and sends different copies of D to A, B and C. There will be a mismatch between at least one pair of these peers, and it will be detected.

    If $d > z_2$, D is fully exchanged between node A and C. A (C) will see a mismatch if S is faulty and sends two different D's to A and C, or if C (A) is faulty and sends some bad bit of D to A (C). so their copies of D must be identical if S is faulty, or A (C) is faulty but sends the correct copy to C (A). If neither A nor C detects a mismatch for packet D, then B is able to check every bit of D with A and C jointly. If A or C is faulty and sends bad bits of D to B, B will detect it. If B is faulty and sends bad bits of D to A or C, A or C will detect it. If S is faulty and sends bad bits to B that are different from what it sends to A and C, at least B or on of A and C will detect it.

- If $p+q+r+u < 2-k$
  - Packets $E, F_{d-\delta}, G$: they are checked in the same way as in the previous case.
  - Packet $D_{d-\delta}$: it is checked in the same way as D in the previous case.
  - Packets D_δ, F_δ: remember that D_δ is compared between A and C, so any misbehavior on it between A and C will be detected by at least one of them. Similarly, F_δ is compared between B and C, so any misbehavior on it will be detected by B or C. A receives D_δ from S, $D_\delta \oplus F_\delta$ from B, and F_δ from C, which form a (3,2) error detection code, and A can detect any single attack by B or C. If S is bad, it can only attack the D_δ B gets without being detected, otherwise A or C will see a mismatch in their D_δ, or B or C will see a mismatch in their F_δ. But any attack by on the D_δ B gets will be detected by the (3,2) code at A.

4.2.3 Extended Round 3 after detection

Once any node detects a "mismatch" in round 3, every node is required to broadcast all the information it has sent and received during round 1 and 2 to the other three nodes, using the traditional Byzantine agreement algorithm, such as the algorithm by Pease, Shostak and Lamport [2], [1]. Since at most one node is faulty, all nodes obtain identical information about what every node "claims" it has sent and received during round 1 and 2. Then each non-faulty peer will compare the information they received during round 1 and 2 with the one they received during the broadcast.

Our discussion below shows that after the broadcast phase, the location of the failure can be narrowed down to one of the following cases: a set containing two peers (Mode II); or a set containing a peer and the source (Mode III); or one node (Mode IV).

- Packet D: In our algorithm, every bit of packet D is compared by at least two pairs of peers. So if a peer, say A, sends some bad bits of D in round 2, and sends the same bad bits of D during the broadcast, it will be detected by B and C both. Then B and C will tell this to each other. This looks the same to B and C as the case when S sends consistent copies of D to B and C, and a bad D to A, they can determine that the faulty node is either A or S (mode II). In case A sends a bad D to B and a good D to C in round 2, if A broadcasts a bad D (may not be the same one it sent to B), every other node will detect it and knows that A is faulty (mode IV). If A broadcasts the good D, it looks the same to C and S as the case B is bad and blames A. Then C and S can be sure that either A or B is faulty (mode III). If S is bad, and it sends three different D's to A, B and C, after the broadcast, all peers see three different D's and can determine that S is faulty (mode IV). Similar situation if B or C is faulty and sends bad D.

- Packet E: C compares E received from A and B in round 2. If A sends a bad E in round 2, and sends the same bad E during the broadcast, B and S will detect it, then B announces that one of S and A is faulty, S announces that A is faulty, and C announces that one of S, A and B is faulty. Then no matter what A announces, S and A receive more "votes" than B and C. Then they can determine that the faulty node is either A or S (mode II). If A broadcasts a good E, C knows A is faulty, and S and B can be sure that either A or C is faulty (mode III). If A broadcasts anything else, S, B, and C will all know A is faulty (mode IV).

  Similar if B is faulty.

  If S is faulty in the first place and sends different E's to A and B in round 1, it can blame A (or B) to be bad after the broadcast. To the other nodes, it looks the same as A being bad, so they can determine that either S or A is faulty (mode III). It is similar if B is being blamed by S.

If C is faulty and raises an false alarm. It has to broadcast two different copies of E, otherwise every other node knows it raises an false alarm and determines C is bad (mode IV). But if C claims in the broadcast that it has received a bad E from A and a good E from B, A knows C is bad, and it looks the same to S and B as if A is faulty and broadcasts a correct E, so they can determine that either A or C is bad (mode II).

- Packets F, G: similar to packet E.

- Packets $H, I, H \oplus I$:

  If A is faulty, it can: (1) send H' to B and C, and broadcast a different H'', it will be detected by both B and C (mode IV); (2) send H' to B and C, and broadcast the same H', S will detect it and accuses A, then B and C determine that either S or A is bad (mode III); (3) send H' to B and H'' to C, and broadcast a different H''' (may be the good H), both B and C will know A is bad (mode IV); (4) send H' to B and H'' to C, and broadcast H', then S and C know A is bad, then B will agree with S and C and determine A is bad (mode IV); (5) send H' to B and H'' to C, and broadcast H'', then S and B know A is bad and C agrees with them and determine A is bad (mode IV).

  It is similar if B or C is faulty.

  If S is faulty, it sends $H_A, I_B, (H \oplus I)_C$ to A, B and C such that $H_A \oplus I_B \neq (H \oplus I)_C$, it must accuse one of the peers after the broadcast. If S accuses A, A knows S is bad, and it looks the same to B and C as if A is bad and send the same bad H during round 2 and the broadcast, so B and C will determine that either S or A is bad (mode III). It is similar if S accuses B or C.

- Packets $D_\delta, F_\delta, D_\delta \oplus F_\delta$ when $p + q + r + u < 2 - k$: similar to $H, I, H \oplus I$.

Now we have shown that once any node misbehaves, the identity of the faulty node is narrowed down to at most two nodes, one of which is indeed faulty, after the extended round 3. Then the system enters mode II, III, or IV accordingly. In mode II and III, the good node among the set knows the faulty node exactly.

How much time should be added to round 3? We assume that the capacity of all link is > 0, but the capacity may be arbitrarily small. Since a finite number of bits need to be exchanged during round 3, with a non-zero capacity, the duration required will be finite as well. As we will see later, the broadcast occurs only once for each mode transition. Since the mode index only increases, the broadcast occurs for at most 3 times. And each broadcast only requires a finite amount of time, so as we will see later, in a pipelined execution, the overhead of round 3 amortized over duration t approaches 0 as $t \to \infty$.

4.3 Operation in Mode II

Without loss of generality, assume that the location of the faulty node is narrowed down to between node A and B, and A is in fact the faulty node. Then B is sure that A is bad. The schedule for round 1 and 2 are the same as in Mode I. Since now B knows A is bad, B can recover all the data from the packets it receives from S and C. Node C trust all the data it receives from S. For the data S does not send to C, C receives one copy from B, and another one from A. If these two copies match, C agrees on that. If not, C detects an mismatch and demands a broadcast in an extended round 3 as in mode I. In this case, A has sent bad packets to C in round 2. So no matter what A broadcasts, both S and C will detect that A is bad and then all good nodes have decided A is faulty (mode IV). Then C will recover the data from the packets it receives from S and B from now on.

4.4 Operation in Mode III

Without loss of generality, assume that the location of the faulty node is narrowed down to between S and A. The schedule in round 1 is the same as in Mode I. The schedule in round 2 and round 3 may need to modified slightly.

If $s+t \geq 1$, there is at least 1 unit of capacity on links BC and CB. B and C exchange a complete copy of the data on BC and CB during round 2. Then B and C both will agree on the data they have exchanged if they do not see a mismatch. In the case B or C sees a mismatch, they can determine that S is faulty (mode IV) immediately. Since A has incoming rate of at least 1 from B and C together, B and C are able to send all the data they have agreed on to A in round 3, jointly. If A is good, it agrees with the data from B and C and agreement among the peers are achieved.

If $s+t = 1-\delta < 1$, first notice that

$$1 - l + 1 - m = 2 - l - m \leq s + t = 1 - \delta, \tag{42}$$

hence

$$l + m \geq 1 + \delta, \tag{43}$$

which implies that at least δ unit of data are sent to both B and C. B and C can exchange the other $1-\delta$ unit of data on links BC and CB directly in round 2, and agree on that if there is no mismatch. Otherwise they can determine that S is faulty (mode IV) immediately. Then in round 3, B and C send the $1-\delta$ unit of data they already agreed on to A. Remember that A has incoming rate from B and C together of at least 1, there is at least δ unit of capacity left on BA and CA together. Schedule part of the δ unit of data that S has sent to both B and C on BA and the rest on CA in round 3 such that the capacity of these two links is not exceeded. Now notice that r (AB) and u (AC) are both at least δ, since $s \leq 1-\delta$

and $t \leq 1 - \delta$. A is able to send this δ unit of data to both B and C on AB and AC. B and C compare this δ unit of data they receive from A with the one they has received from S in round 1. If they match, agree on that. If not, either S is faulty by sending different δ unit of data to B and C, or A is faulty by forwarding some bad data for B and C. Then the broadcast is performed for the second time, and B and C can determine which one of S and A is faulty (mode IV).

4.5 Operation in Mode IV

Once the network enters mode IV, the identity of the faulty node has been determined. If S is faulty, the peers can terminate the algorithm, or agree on some default value. If a peer is faulty, the schedule is the same as in mode I, and a good peer will recover the data from the packet received from the source and the other good peer.

4.6 Throughput Analysis

Observe that, with the exception of the determination and dissemination of faulty assessments (in round 3), each link is used in only 1 round of the algorithm. Also, ignoring the overhead of computing and disseminating the assessments, each link is used for 1 time unit over all the rounds combined (this conclusion follows by adding up the size of packets sent on each link during all rounds). The dissemination of assessments (in round 3) requires a fixed number of bits per link, independent of R. Thus, the total time overhead for this operation is $O(1/R)$, and by increasing R (which can be achieved by scaling the unit of time), it is possible to make the overhead of diagnosis messages arbitrarily close to zero. This allows us to achieve throughput arbitrarily close to 1, as follows.

In achieving rate close to 1, it will be necessary to have multiple "generations" of packets in the network, with the algorithm operating in a pipelined manner (one round per pipeline stage). Agreement algorithm for one new generation of data of size 1 unit starts per clock cycle (where the clock cycle of the pipeline is long enough for each of the rounds), as shown in Figure 4. Each generation consists of three rounds and the packets are scheduled according to the schedules we discussed in the previous section, depending on which mode the system is in. At the beginning of the round 3 of every generation, the peers exchange 1 bit indicating whether they detect a mismatch. At the end of the round 3 of every generation, the peers exchange their own assessments and determine the new mode of the system. The time duration required for data transmission in each round is at most 1 time unit (recall that no link carries more information packets than its capacity permits). The assessment and dissemination operations require $O(1/R)$ time, which can be made small by choosing a large R.

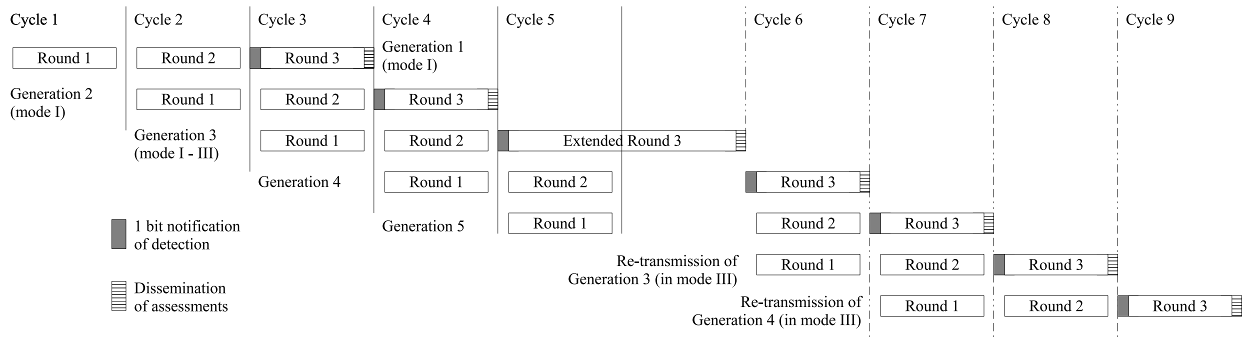
Cycle 1
Cycle 2
Cycle 3
Cycle 4
Cycle 5
Cycle 6
Cycle 7
Cycle 8
Cycle 9
Round 1
Round 2
Round 3
Generation 1 (mode I)
Generation 2 (mode I)
Generation 3 (mode I - III)
Generation 4
Generation 5
Extended Round 3
1 bit notification of detection
Dissemination of assessments
Re-transmission of Generation 3 (in mode III)
Re-transmission of Generation 4 (in mode III)

Figure 4: Example of pipelining: In generation 2, node A detects a mismatch and the system enters mode III. Generations 3, 4 and 5 are then dropped and retransmitted after entering mode III.

Thus, we can make the duration of each round to be equal to $1 + O(1/R)$. Since a new generation of 1 unit worth of information is initiated in each round, it follows that the agreement throughput is $1 - O(1/R)$. Thus, by scaling the time unit, the throughput can be made arbitrarily close to 1.

Figure 5 shows an example execution of the pipelining. The system starts in mode I. Nominally 1 clock cycle is assigned to each round, including round 3. When round 3 is not extended, much of the time in round 3 would be idle time. The system is in mode I for generations 1 and 2. During round 3 of generation 3, node A detects a mismatch and require the system enter the broadcast phase, and the round 3 is extended as shown. By the end of this round (cycle 5), the identity of the faulty node is confined to A,S and the system enters mode III. Note that, at the time the system enters mode III, three generations of packets are in the system using the old schedule (in this example: generations 3, 4 and 5). To allow a transition to the new schedule in mode III, the packets belonging to generations 3, 4 and 5 are dropped, and retransmitted using the algorithm/schedule for mode III. Thus, agreement for the dropped generations is re-initiated in subsequent clock cycles.

Observe that with the above pipelined operation, each link is used for data transmission at most once in each round, no matter how the mode changes. This implies that all the links are being used within their capacity constraint. Although three generations may be dropped when the system enters mode III, the overhead of dropped generations approaches zero asymptotically (as time t approaches ∞). Recall that the transition to mode III occurs only once. Similarly, the constant duration overhead (constant, independent of R) of extended round 3 also occurs only a finite number of times - thus, the normalized overhead of extended round 3 decreases as R increases. Thus, the normalized throughput approaches 1 asymptotically despite the dropped generations, as $t \to \infty$ and $R \to \infty$.

5 An Alternative Proof of Sufficiency

The sufficiency of the same set of conditions can also be proven by applying a more general linear codes at the source (Reed-Solomon codes, for example). The algorithm operates in the same four modes we discussed before.

5.1 Operation in Mode I

The R units of data is divided into R unit-size packets, and each packet is expressed as a vector. The source generates $k+l+m$ linear combinations of the packets. With a sufficiently large field size, any set of R of these equations can be made linear independent. If none of the equations are corrupted, any R of them are uniquely solved by the vector x, which represents the original R units of data.

Let's index these equations from 1 to $k + l + m$. The equations are transmitted according to the following schedule.

Round 1 : S transmits equations 1 to k to A, $k+1$ to $k+l$ to B, and $k+l+1$ to $k+l+m$ C.

Round 2 : Each peer sends as many equations it receives from S in round 1 as possible to the other two peers, in the increasing order of index. In other words, A sends equations 1 to $\min\{k, r\}$ to B, and equations 1 to $\min\{k, u\}$ to C; B sends equations $k+1$ to $k+\min\{l, p\}$ to A, and equations $k+1$ to $k+\min\{l, t\}$ to C; C sends equations $k+l+1$ to $k+l+\min\{m, q\}$ to A, and equations $k+l+1$ to $k+l+\min\{m, s\}$ to B.

Round 3 : Each peer tries to find a unique solution for all the equations it has received during round 1 and 2. If there is no such unique solution, the peer detects an attack and the system enters the broadcast phase. During the broadcast phase, round 3 is extended as in our earlier algorithm, and every node is required to broadcast all the equations it has sent and received during round 1 and 2. After the broadcast phase, the system enters mode II, III, or IV, depending on the outcome of the assessment.

Now we will show that any misbehavior will be detected by at least one peer node in round 3.

If a peer node is faulty, say node A, it can either attack by raising a false alarm or sending some bad equations to the other two peers. If it raises a false alarm, a failure is already detected. In the latter, suppose A sends some bad equations to B. If these bad equations can be satisfied by x, then they are actually not bad, since x is still the unique solution to all the equations B receives. If these bad equations cannot be satisfied by x, B cannot find any solution to all the equations it has, since B receives at least R correct equations from S and C together, which are solved uniquely by x, but x does not satisfy the bad equations from A. Then B will detect the attack.

If S is faulty, it can either try to make two peers decide on the same vector and the third peer decide on a different vector, or try to make three peers decide on three different vectors.

In the first case, suppose S tries to make A and B decide on x and C decide on x'. Then A's and B's equations are all satisfied by x and some of C's equations must not be satisfied by x, otherwise all of C's equations are satisfied by x and C will also decide on x, as if S is not misbehaving. Since C receives R equations from A and B, which is solved uniquely by x, C cannot find any solution to all the equations it receives, and detects the attack.

In the latter case, suppose S tries to make A decide on x, B decide on $x' \neq x$, and C decide on $x'' \neq x, x'$. Then at least one peer will detect the attack. Suppose in contradiction that none of the peers can detect. Denote $E_{i,j}$ as the equations node i sends to node j. Remember that from condition 11 to 13, $|E_{A,B}|+|E_{B,A}|+|E_{B,C}|+|E_{C,B}|+|E_{A,C}|+|E_{C,A}| \geq 3R$. It is not hard to see that at least one of $|E_{A,B}|+|E_{B,A}|$, $|E_{B,C}|+|E_{C,B}|$, $|E_{A,C}|+|E_{C,A}|$ has size of at least R. Without loss of generality, assume $|E_{A,B}|+|E_{B,A}| \geq R$. By assumption, A cannot detect the attack, $\{E_{S,A}, E_{B,A}\}$ are uniquely solved by x. And since $E_{A,B} \subseteq E_{S,A}$ and $|E_{A,B}| + |E_{B,A}| \geq R$, $\{E_{A,B}, E_{B,A}\}$ are uniquely solved by x too. On the other hand, since B cannot detect the attack and decides on x', $\{E_{S,B}, E_{A,B}\}$ are uniquely solved by $x' \neq x$. Since $E_{B,A} \subseteq E_{S,B}$ and $|E_{A,B}| + |E_{B,A}| \geq R$, $\{E_{A,B}, E_{B,A}\}$ are uniquely solved by x' as well. Then $\{E_{A,B}, E_{B,A}\}$ are solved by both x and x', which implies $x = x'$ and contradicts with our assumption.

The proof that the location of the faulty node can be narrowed down to at most two nodes after the broadcast phase is similar to that of the deterministic algorithm. It is omitted here to avoid redundancy.

5.2 Operations in Mode II, III, and IV

We are not going to discuss the details of the operations in mode II, III, and IV since they are similar to those in the deterministic algorithm.

6 General Networks

In the previous sections, we have shown the following two conditions

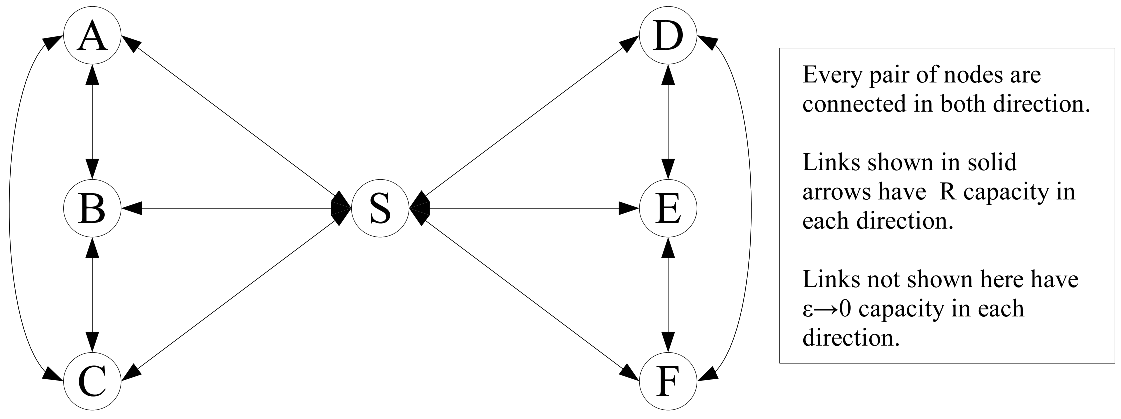
A
B
C
S
D
E
F
Every pair of nodes are connected in both direction.
Links shown in solid arrows have R capacity in each direction.
Links not shown here have ε→0 capacity in each direction.

Figure 5: Every link in this network has capacity R in each direction. The two conditions sufficient for the four node network are not sufficient for the seven node network.

- When one of the peers is removed from the network, the min-cut from the source to any remaining peer is greater than R.
- The max-flow to each of the peers from the other peers, with the source removed, at greater than R.

is sufficient for agreement throughput R bits/unit time to be achievable for the four node network. Since these sufficient conditions can be arbitrarily close to the necessary conditions in Section 2, we have actually characterized the capacity of Byzantine agreement of the four node network. However, as we are going to show, these conditions are not sufficient for large general network topologies.

Consider the seven node network shown in Figure 5. Each link in this network has capacity R in each direction. We can see that the two conditions are both satisfied:

- After removing any one of the peers, the min-cut from the source to any remaining peer is at least $2R$.
- After removing the source, the max flow going into any one of the peers is at least $2R$.

However, it is easy to see that the Byzantine agreement capacity of this network is zero. The adversary can attack S and make A, B, C agree on x and D, E, F on a different x'.

7 Conclusion

In this report, we provide the tight bound of the capacity of Byzantine agreement of the four node network. Many problems remain open, including generalization of the agreement algorithm to other topologies and larger number of failures, and a complete capacity characterization.

References

[1] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *JOURNAL OF THE ACM*, vol. 27, pp. 228–234, 1980.

[2] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, pp. 382–401, 1982.

[3] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger, "Byzantine modification detection in multicast networks using randomized network coding," 2004.

[4] H. Attiya and J. Welch, *Distributed Computing.* McGraw-Hill, 1998.

[5] N. Vaidya and G. Liang, "Capacity of byzantine agreement (preliminary draft - work in progress)," *Technical Report, CSL, UIUC*, January 2010.