

# Interplay between (Im)perfectness, Synchrony and Connectivity: The Case of Reliable Message Transmission

**Abhinav Mehta<sup>†</sup>, Shashank Agrawal<sup>†</sup>, Kannan Srinathan<sup>†</sup>**  
{abhinav\_mehta,shashank.agrawal}@research.,srinathan@}iiit.ac.in

**ABSTRACT.** For unconditionally reliable message transmission (URMT) in synchronous directed networks of  $n$  nodes, a subset of which may be Byzantine faulty, it is well-known that the minimum connectivity requirements for zero-error (perfect) protocols to exist is strictly higher than those where a negligible yet non-zero error probability is allowed (*Monte Carlo* protocols) [9]. In this work, we study the minimum connectivity requirements for the existence of (a) *Las Vegas* protocols in synchronous networks, (b) asynchronous *Monte Carlo* protocols and (c) asynchronous *Las Vegas* protocols for URMT.

Interestingly, we prove that in any network, synchronous *Las Vegas* URMT protocol exists if and only if asynchronous *Monte Carlo* URMT protocol exists too. We further show another interesting result that asynchronous *Las Vegas* URMT protocols exist if and only if synchronous perfect protocols exist. In a nutshell, our results establish the following hierarchy with respect to the connectivity requirements for URMT protocols (in the following, P stands for perfect, MC stands for *Monte Carlo*, LV stands for *Las Vegas*, S denotes synchronous and A denotes asynchronous; it is known that  $SMC < SP = AP$ ):

$$SMC < SLV = AMC < ALV = SP = AP$$

## 1 Introduction

In the unconditionally reliable message transmission (URMT) problem, two non-faulty players, the sender  $\mathbf{S}$  and the receiver  $\mathbf{R}$  are part of a communication network modelled as a digraph over  $n$  players/nodes influenced by an unbounded active adversary that may corrupt some subset of these  $n$  players/nodes.  $\mathbf{S}$  has a message that he wishes to send to  $\mathbf{R}$ ; the challenge is to design a protocol such that  $\mathbf{R}$  correctly obtains  $\mathbf{S}$ 's message with arbitrarily small error probability, irrespective of what the adversary (maliciously) does to disrupt the protocol. Note that by “unconditional”, we mean that the adversary is of unbounded computational power and therefore modern cryptographic tools for verifying the integrity of the data are irrelevant.

Analogous to randomized sequential algorithms, one may distinguish between two variants of URMT, namely, *Monte Carlo* and *Las Vegas*. In the former variant  $\mathbf{R}$  outputs the sender's message with high probability and may produce an incorrect output with small probability; in the latter,  $\mathbf{R}$  outputs the sender's message with high probability and with small probability may abort the protocol but in no case does the receiver terminate with an incorrect output. While *Monte Carlo* URMT has been studied in [8, 9], we initiate the study of *Las Vegas* URMT over directed synchronous networks and characterize the exact gap in

---

<sup>†</sup>Center for Security, Theory and Algorithmic Research (C-STAR),  
International Institute of Information Technology, Hyderabad, 500032, India.

the class of networks over which Las Vegas URMT, as compared to Monte Carlo URMT, is possible.

Most of the distributed computing protocols assume that every pair of participating nodes share a reliable channel which with small probability may abort the protocol but in no case does the receiver terminate with an incorrect output. While Monte Carlo URMT has been studied in [8], we initiate the study of Las Vegas URMT over directed synchronous networks and characterize the exact gap in the class of networks over which Las Vegas URMT, as compared to Monte Carlo URMT, is possible.

We also initiate the study of Monte Carlo URMT protocols over *asynchronous* directed networks. Unlike *synchronous* networks, in which the players have full information about the timings of the events in the network, in an asynchronous network, a conservative and more realistic assumption is used, namely that no time-bounds are known to the players regarding the schedule of various events in the network. Clearly, Monte Carlo URMT over asynchronous digraphs is harder to achieve (and indeed requires more network connectivity) than Monte Carlo URMT over synchronous digraphs. Equally evident is the fact that achieving Las Vegas URMT is harder (and again it indeed requires more network connectivity) than achieving Monte Carlo URMT, over synchronous digraphs. Though not seemingly related, interestingly, we prove that the additional requirements in network connectivity in both the aforementioned cases is exactly the same.

In the sequel, we similarly study the minimum connectivity requirements for the existence of asynchronous Las Vegas URMT protocols, which interestingly turns out to be the same as those for the existence of synchronous perfect protocols.

We further improve our insights in the problem by studying on how sparse can a digraph that permits URMT be. Specifically, we say that an edge is *critical* if its removal renders the graph insufficiently connected for URMT protocols (though before its removal the connectivity was sufficient). Ironically, it turns out that for perfect protocols, the number of critical edges is always  $O(n)$  where as for the “easier” randomized protocols, we give a family of digraphs with  $\Omega(n^2)$  critical edges! We remark that an earlier attempt in [8] to give such a family of digraphs for the case of synchronous Monte Carlo URMT protocols is incorrect and we correct the same; we also give similar families of digraphs (with  $\Omega(n^2)$  critical edges) for synchronous Las Vegas (and asynchronous Monte Carlo) protocols.

## 1.1 Related Work

The problem of URMT was first defined by Franklin et. al. in [4]. They consider the problem of Monte Carlo URMT over undirected networks tolerating *threshold* Byzantine adversary and show that URMT is possible iff PRMT is possible. In [2], Desmedt et. al. give efficient protocols for URMT (which also achieve perfect privacy) in directed networks by abstracting the network as a collection of disjoint wires between **S** and **R**.

A more general setting is considered in [9, 8], which is also the setting with which we work in this paper, where the underlying network is abstracted as a directed graph and every node is as powerful as probabilistic Turing Machine. Such an abstraction is well motivated in practice because not every communication channel admits bi-directional communication and intermediate nodes are usually not mere routers, but can compute too. The

characterization of directed synchronous networks for the possibility of Monte Carlo URMT tolerating *non-threshold mixed adversary* is done in [9], a simplified characterization in the case of threshold Byzantine adversary follows in [8].

The theorem establishing equivalence between the connectivity requirements for the possibility of synchronous Las Vegas URMT and that of asynchronous Monte Carlo URMT appears without proof as a brief announcement in the proceedings of DISC 2010 [6]. Detailed proofs of this result appear in this paper (alongwith several other important results).

## 1.2 Organization of paper

We characterize directed networks for the possibility of (i) Las Vegas variant of URMT in synchronous networks (ii) Monte Carlo variant of URMT in asynchronous networks (iii) Las Vegas variant of URMT in asynchronous networks, tolerating a non-threshold Byzantine adversary  $\mathbb{A}$  in Section 3, 4 and 5 respectively. Dealing with arbitrary sized adversary structure is hard and non-intuitive, so we take the following mentioned two step approach in each of these sections. We first prove that in order to tolerate an adversary structure  $\mathbb{A}$  it is sufficient to tolerate all its two sized subsets. We then give a necessary and sufficient condition for the respective variant of URMT to be possible tolerating a two-sized adversary structure. In Section 6, we study the concept of *critical edges*. In Theorem 31, we prove that the family of digraphs over  $n$  nodes proposed in [8] for synchronous Monte Carlo URMT claimed to have  $\Omega(n^2)$  critical edges has only  $O(n)$  critical edges. Theorem 32 presents another family which has indeed  $\Omega(n^2)$  critical edges. Further, in Theorem 33, we give a family of digraphs with  $\Omega(n^2)$  critical edges w.r.t. the two variants of URMT studied in this paper - synchronous Las Vegas and asynchronous Monte Carlo.

## 2 Model and Definitions

We model the underlying network in which a sender  $\mathbf{S}$  and a receiver  $\mathbf{R}$  are two distinguished nodes as a directed graph  $\mathcal{N} = (V, \mathcal{E})$ , where  $V$  is the set of nodes and  $\mathcal{E} \subseteq V \times V$  is the set of all directed edges in the network. We assume the secure channels setting, i.e., all the edges are secure, reliable and authenticated. We assume that every node is a probabilistic Turing Machine and the topology of the network is known to every node.

Following [5], we model faults in the network by a fictitious centralized entity called the *adversary* which has unbounded computing power. A single “snapshot” of faults in the network can be described as  $B \subseteq V \setminus \{\mathbf{S}, \mathbf{R}\}^*$  which means that the nodes in the set  $B$  are faulty. We denote the set of all such  $B$ 's by  $\mathbb{A}$  and refer to it as an *adversary structure*. We allow Byzantine corruption, i.e., all nodes in the set  $B \in \mathbb{A}$  corrupted by the adversary can deviate arbitrarily from the designated protocol. Sometimes the adversary may only *fail-stop* a corrupted node in which case the node stops sending and receiving any messages.

The adversary structure is *monotone*: if  $B_1 \in \mathbb{A}$  then  $\forall B_2 \subset B_1, B_2 \in \mathbb{A}$ . We note that  $\mathbb{A}$  can be uniquely represented by listing the elements in its *maximal basis*  $\overline{\mathbb{A}} = \{B \mid B \in \mathbb{A}, \nexists X \in \mathbb{A} \text{ s.t. } B \subset X\}$ . Abusing the standard notation, we assume that  $\mathbb{A}$  itself is a

---

\*Problem of reliable communication makes sense only when the sending node  $\mathbf{S}$  and the receiving node  $\mathbf{R}$  are non-faulty, for otherwise reliable communication need not happen.

maximal basis. We refer to  $\mathbb{A}$  as a  $t$ -threshold adversary if  $\mathbb{A} = \{B \mid B \subseteq V \setminus \{\mathbf{S}, \mathbf{R}\} \text{ and } |B| = t\}$ . We only deal with cases where  $|\mathbb{A}| \geq 2$ , since otherwise the problems are trivial.

We allow the adversary to be *adaptive* – it can choose which nodes to corrupt during the execution of a protocol based on its view, as long as the set of nodes corrupted during the entire execution is a member of  $\mathbb{A}$ . We assume that the adversary knows the topology of the network as well as the protocol specification. We further assume that it knows the message sender  $\mathbf{S}$  has chosen to send to  $\mathbf{R}$ <sup>†</sup>.

Protocols running over directed networks tolerating an adversary rely on the information of timing of the events in the system. We consider two extremes w.r.t timing model, i.e., either all the edges in the network are *synchronous* or *asynchronous*. In the former a protocol is executed in a sequence of *rounds* where in each round, a player can send messages to his out-neighbours, receive the messages sent in that round by his in-neighbours and perform some computation on the received messages, in that order. In the latter case (when the network is asynchronous), there is no fixed upper bound on the timings of events. To model asynchrony in the network we assume that the adversary is additionally equipped with the ability to schedule all the messages exchanged over the network while remaining oblivious to the messages being exchanged [1].

Let the message space be a large finite field  $\langle \mathbb{F}, +, \cdot \rangle$ . All the computations are done in this field. In this paper we refer to *Las Vegas* URMT as  $\text{URMT}_{LV}$  and *Monte Carlo* URMT as  $\text{URMT}_{MC}$ . We may also use URMT without any subscript to refer to both the variants together. We now formally define what we mean by a protocol being  $\text{URMT}_{LV}$  or  $\text{URMT}_{MC}$  protocol. All the probabilities are taken over the coin tosses of all the nodes and the adversary.

**DEFINITION 1.** [ $(\mathbb{A}, \delta)$ - $\text{URMT}_{MC}$ ] Let  $\delta < \frac{1}{2}$ . We say that a protocol for transmitting messages in a network  $\mathcal{N}$  from  $\mathbf{S}$  to  $\mathbf{R}$  is  $(\mathbb{A}, \delta)$ - $\text{URMT}_{MC}$  if for all valid Byzantine corruptions of any  $B \in \mathbb{A}$  and  $\forall \mathbf{m} \in \mathbb{F}$ , the probability that  $\mathbf{R}$  outputs  $\mathbf{m}$  given that  $\mathbf{S}$  has sent  $\mathbf{m}$ , is at least  $(1 - \delta)$ . Otherwise  $\mathbf{R}$  outputs  $\mathbf{m}' \neq \mathbf{m}$  or does not terminate. If the network is synchronous,  $\mathbf{R}$  is bound to terminate with certainty.

**DEFINITION 2.** [ $(\mathbb{A}, \delta)$ - $\text{URMT}_{LV}$ ] Let  $\delta < \frac{1}{2}$ . We say that a protocol for transmitting messages in a network  $\mathcal{N}$  from  $\mathbf{S}$  to  $\mathbf{R}$  is  $(\mathbb{A}, \delta)$ - $\text{URMT}_{LV}$  if for all valid Byzantine corruptions of any  $B \in \mathbb{A}$  and  $\forall \mathbf{m} \in \mathbb{F}$ , the probability that  $\mathbf{R}$  outputs  $\mathbf{m}$  given that  $\mathbf{S}$  has sent  $\mathbf{m}$ , is at least  $(1 - \delta)$ . Otherwise,  $\mathbf{R}$  outputs a special symbol  $\perp$  ( $\notin \mathbb{F}$ ) or does not terminate.

**DEFINITION 3.** [ $\mathbb{A}$ -PRMT] We say that a protocol for transmitting messages in a network  $\mathcal{N}$  from  $\mathbf{S}$  to  $\mathbf{R}$  is  $\mathbb{A}$ -PRMT if for all valid Byzantine corruptions of any  $B \in \mathbb{A}$  and  $\forall m \in \mathbb{F}$ , the probability that  $\mathbf{R}$  outputs  $\mathbf{m}$  when  $\mathbf{S}$  has sent  $\mathbf{m}$  is 1.

We refer to  $(\mathbb{A}, \delta)$ -URMT (resp.  $\mathbb{A}$ -PRMT) as a  $(t, \delta)$ -URMT (resp.  $t$ -PRMT) when  $\mathbb{A}$  is a  $t$ -threshold adversary.

---

<sup>†</sup>If we do not make this assumption, the results we prove in this paper still hold but with a slight change in our definition of URMT (see [4]).

**DEFINITION 4.**[Strong path] A sequence of vertices  $v_1, v_2, v_3, \dots, v_k$  is said to be a strong path from  $v_1$  to  $v_k$  in the network  $\mathcal{N} = (V, \mathcal{E})$  if for each  $1 \leq i < k$ ,  $(v_i, v_{i+1}) \in \mathcal{E}$ . Furthermore, we assume that there vacuously exists a strong path from a node to itself.

**DEFINITION 5.**[Weak path] A sequence of vertices  $v_1, v_2, v_3, \dots, v_k$  is said to be a weak path from  $v_1$  to  $v_k$  in the network  $\mathcal{N} = (V, \mathcal{E})$  if for each  $1 \leq i < k$ ,  $(v_i, v_{i+1}) \in \mathcal{E}$  or  $(v_{i+1}, v_i) \in \mathcal{E}$ .

**DEFINITION 6.**[Blocked node] A node  $u$  along a weak path  $p$  is called a blocked node if its out-degree along  $p$  is 0.

**DEFINITION 7.**[Head node] A node  $y$  along a weak path  $p$  is called a head node if it is an intermediate node with out-degree 2 or a terminal node with out-degree 1.

Every weak path  $p$  between  $\mathbf{S}$  and  $\mathbf{R}$  can be viewed as an alternating sequence of blocked nodes  $u_i$ 's and head nodes  $y_i$ 's starting with  $\mathbf{S}$  as a head node denoted by  $y_0$  and ending into  $\mathbf{R}$  as a blocked denoted by  $u_{n+1}$  i.e.  $\exists n > 0$ , path  $p$  can be viewed as  $y_0, u_1, y_1, u_2, y_2, \dots, u_n, y_n, u_{n+1}$  such that each  $y_i$  has a strong path to  $u_i$  and  $u_{i+1}$  along  $p$ . The case when  $\mathbf{S}$  is not a head node (i.e. it is a blocked node) along  $p$ , the following described simulation within the node  $\mathbf{S}$  can ensure that a virtual sender  $\mathbf{s}$  is a head node: (i)  $\mathbf{S}$  simulates two nodes,  $\mathbf{s}$  and  $\mathbf{u}$  and a directed edge  $(\mathbf{s}, \mathbf{u})$ . (ii) The incoming path to  $\mathbf{S}$  along  $p$  (which made  $\mathbf{S}$  a blocked node) is now an incoming path to  $\mathbf{u}$ .

Analogously, it can always be ensured that  $\mathbf{R}$  is a blocked node. We elaborate our representation of weak path with an example in Figure 1. Such a representation of a weak path comes handy in giving easy to understand sufficiency proofs.

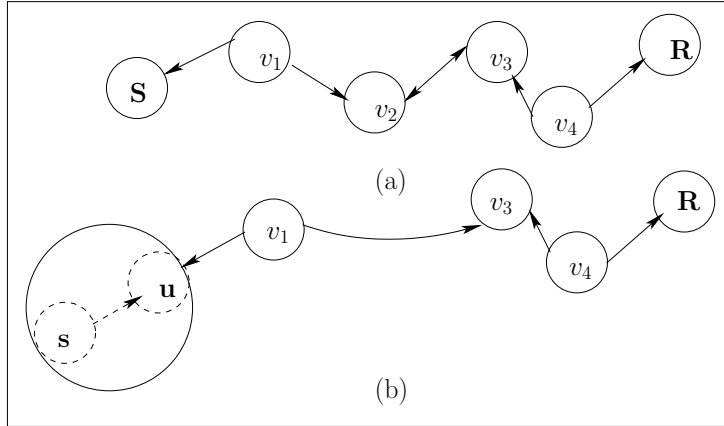


Figure 1: (a) A weak path between  $\mathbf{S}$  and  $\mathbf{R}$ . (b) We view it as a sequence of alternating blocked nodes and head nodes starting with a head node  $\mathbf{s}$  (the virtual sender) and ending in a blocked node  $\mathbf{R}$ . Nodes  $\mathbf{s}$ ,  $v_1$  and  $v_4$  are head nodes;  $\mathbf{u}$  and  $v_3$  are blocked nodes such that there is a strong path from  $\mathbf{s}$  to  $\mathbf{u}$ ;  $v_1$  to  $\mathbf{u}$  and  $v_3$ ; and  $v_4$  to  $v_3$  and  $\mathbf{R}$ .

**DEFINITION 8.**[Critical edge] In a digraph  $G$  for which URMT protocol exists, an edge is said to be critical if the deletion of that edge renders URMT impossible.

**DEFINITION 9.**[Authentication function] Let  $K_1, K_2, K_3 \in_R \mathbb{F} \times \mathbb{F} \times \mathbb{F}$  and  $m \in \mathbb{F}$ . Authentication function  $\chi$  is defined as  $\chi(m; K_1, K_2, K_3) = (m + K_1, (m + K_1) \cdot K_2 + K_3)$ .

The authentication function *authenticates* any message  $m \in \mathbb{F}$ . We refer to  $K_1, K_2$  and  $K_3$  as *keys*. If three randomly chosen keys (unknown to the adversary) are established between two nodes  $u$  and  $v$  such that  $\exists$  a path  $p$  from  $u$  to  $v$  then authentication function is used as follows: (a) Say,  $x = m + K_1$  and  $y = x \cdot K_2 + K_3$ ;  $u$  sends  $\langle x, y \rangle$  (a two tuple) to  $v$  along path  $p$ . (b) Say node  $v$  receives  $\langle x', y' \rangle$ .  $v$  verifies if  $y' \stackrel{?}{=} x' \cdot K_1 + K_2$ . If the verification passes then  $x' = x$  with probability at least  $\frac{|\mathbb{F}|-1}{|\mathbb{F}|}$ , or otherwise  $v$  can deduce with certainty that  $p$  is a faulty path. (Proofs for the same appear in [7]).

### 3 Characterizing synchronous networks for $(\mathbb{A}, \delta)$ -URMT<sub>LV</sub>

In the following theorem we show that in the case of synchronous networks, the problem of characterizing networks for (the (im)possibility of)  $(\mathbb{A}, \delta)$ -URMT<sub>LV</sub> reduces to the problem of characterizing networks for  $(B, \delta)$ -URMT<sub>LV</sub>, where  $|B| = 2$  (similar reductions can be found in [9, 8]).

**THEOREM 10.** *In a directed synchronous network  $\mathcal{N}$ ,  $(\mathbb{A}, \delta)$ -URMT<sub>LV</sub> protocol is possible if and only if for every adversary structure  $B \subseteq \mathbb{A}$  such that  $|B| = 2$ ,  $(B, \delta)$ -URMT<sub>LV</sub> protocol is possible.*

**PROOF.** *Necessity:* Obvious. *Sufficiency:* We show how to construct a protocol tolerating an adversary structure of larger size from protocols tolerating adversary structures of smaller size without increasing the probability of error. Therefore if protocols tolerating adversary structures of size two are available, we can inductively construct protocol tolerating any arbitrary sized adversary structure.

Let  $f \in \mathbb{F}$  be any element **S** intends to send to **R**. Let  $\mathcal{A} \subseteq \mathbb{A}$ . Consider three  $\lceil \frac{2|\mathcal{A}|}{3} \rceil$ -sized subsets of  $\mathcal{A}$ , namely  $\mathcal{A}_1, \mathcal{A}_2$  and  $\mathcal{A}_3$ , such that each element of  $\mathcal{A}$  occurs in at least two distinct  $\mathcal{A}_i$ 's. For  $i \in \{1, 2, 3\}$ , let  $Y_i$  be a  $(\mathcal{A}_i, \delta)$ -URMT<sub>LV</sub> protocol. Using  $Y_i$  as sub-protocol, we first construct  $Z_i$  which is a  $(\mathcal{A}, \frac{\delta}{2})$ -URMT<sub>LV</sub> protocol. This is done by repeating each  $Y_i$  sufficiently many times, keeping the **S**'s input same as  $f$ , in order to amplify the probability of success. We then use  $Z_i$ 's to construct a protocol  $\Gamma$  which is an  $(\mathcal{A}, \delta)$ -URMT<sub>LV</sub> protocol (as proved in the following lemma) as follows:

- For each  $i \in \{1, 2, 3\}$ , sub-protocol  $Z_i$  is run on  $f$ .
- **R** outputs the majority of the outcomes of the three sub-protocols and in case there is no majority, it outputs  $\perp$ .

**LEMMA 11.** *For the directed synchronous network  $\mathcal{N}$ , the protocol  $\Gamma$  constructed above is an  $(\mathcal{A}, \delta)$ -URMT<sub>LV</sub> protocol.*

**PROOF.** Any set  $B \in \mathcal{A}$  is present in at least two subsets among  $\mathcal{A}_1, \mathcal{A}_2$  and  $\mathcal{A}_3$ ; say the two subsets are  $\mathcal{A}_2$  and  $\mathcal{A}_3$ . Hence the outcome of the two sub-protocols  $Z_2$  and  $Z_3$  is correct with at least  $1 - \frac{\delta}{2}$  probability each. Since **R** outputs the majority of the outcomes, its output is correct if both the sub-protocols produce the correct outcome which happens



with at least  $(1 - \frac{\delta}{2})^2$  probability. Hence the error probability is upper bounded by  $\delta - \frac{\delta^2}{4}$  or  $\delta$ . Additionally, it is easy to see that  $\mathbf{R}$  would never output an incorrect message.  $\blacksquare$

Having reduced the problem of  $\text{URMT}_{LV}$  in a synchronous network tolerating an adversary structure to the problem of  $\text{URMT}_{LV}$  tolerating all its 2-sized subsets, we now proceed to characterize directed synchronous networks in which  $\text{URMT}_{LV}$  tolerating adversary structure  $B = \{B_1, B_2\}$  is possible (where  $B_1, B_2 \in \mathbb{A}$ ).

**THEOREM 12.** *In a directed synchronous network  $\mathcal{N}$ ,  $(B, \delta)$ - $\text{URMT}_{LV}$  protocol is possible if and only if for each  $\alpha \in \{1, 2\}$ , there exists a weak path  $q_\alpha$  avoiding nodes in  $B_1 \cup B_2$  such that every node  $u$  along the path  $q_\alpha$  has a strong path to  $\mathbf{R}$  avoiding all nodes in  $B_{\bar{\alpha}}^\dagger$  (Paths  $q_1, q_2$  need not be distinct.)*

We prove the theorem in the following sub-sections.

### 3.1 Sufficiency

For a directed synchronous network  $\mathcal{N}$ , which satisfies the conditions given in Theorem 12, we show how to construct a protocol  $\Pi$  tolerating the adversary structure  $B = \{B_1, B_2\}$ . Let  $m$  be the message  $\mathbf{S}$  intends to send. If either  $q_1$  or  $q_2$  is a strong path from  $\mathbf{S}$  to  $\mathbf{R}$ ,  $\mathbf{S}$  trivially sends  $m$  along that path. When this is not the case, we construct two sub-protocols  $\Pi_1$  and  $\Pi_2$ . For each  $i \in \{1, 2\}$ , protocol  $\Pi_i$  uses the honest weak path  $q_i$ . We give a construction for  $\Pi_1$ , and the construction of  $\Pi_2$  follows by symmetry. For convenience of writing the protocol, we first represent weak path  $q_1$  as  $y_0, u_1, y_1, u_2, y_2, \dots, u_n, y_n, u_{n+1}$  as explained in Section 2 (right after the definitions of weak path, head node and blocked node). We denote  $\mathbf{S}$  and  $\mathbf{R}$  interchangeably as  $y_0$  and  $u_{n+1}$  respectively.  $\Pi_1$  proceeds in the following steps:

1.  $\mathbf{S}$  sends  $m$  to  $u_1$  along  $q_1$ . For  $1 \leq k \leq n$ , node  $y_k$  chooses  $3^k$  random keys namely  $K_{k,1}, K_{k,2}, \dots, K_{k,3^k}$  and sends those to  $u_k$  and  $u_{k+1}$ .
2. Node  $u_1$  receives  $m$  from  $\mathbf{S}$  and keys  $K_{1,1}, K_{1,2}, K_{1,3}$  from  $y_1$ . It calculates  $(\psi_{1,1}, \phi_{1,1}) = \chi(m; K_{1,1}, K_{1,2}, K_{1,3})$  and sends those to  $\mathbf{R}$  along a strong path avoiding  $B_2$  in some fixed round  $r_{u_1}$ .

For  $1 < k \leq n$ ,  $u_k$  receives  $3^{k-1}$  keys from  $y_{k-1}$  and  $3^k$  keys from  $y_k$ . It authenticates the keys received from  $y_{k-1}$  with the keys received from  $y_k$  and sends it to  $\mathbf{R}$  along a strong path avoiding  $B_2$  in some fixed round  $r_{u_k}$ . Formally,  $u_k$  calculates,  $\forall j \ 1 \leq j \leq 3^{k-1}$ ,  $(\psi_{k,j}, \phi_{k,j}) = \chi(K_{k-1,j}; K_{k,3j-2}, K_{k,3j-1}, K_{k,3j})$ .

3.  $\mathbf{R}$  receives  $\{K'_{n,1}, K'_{n,2}, \dots, K'_{n,3^n}\}$  from  $y_n$ .  $\mathcal{N}$  being a synchronous network,  $\mathbf{R}$  knows exactly the round number, say  $r'_{u_k}$ , in which it will receive messages that  $u_k$  sent to it in round  $r_{u_k}$ . If  $\mathbf{R}$  does not receive valid messages from  $u_k$  in round  $r'_{u_k}$ , it assumes that  $B_1$  is faulty and stops. Else if it receives  $\forall k \ \forall j \ 1 \leq k \leq n, 1 \leq j \leq 3^{k-1}$ ,  $(\psi'_{k,j}, \phi'_{k,j})$ , the protocol proceeds as follows.

for  $k$  in  $n$  to 2

$\mathbf{R}$  verifies  $\forall j, \phi'_{k,j} \stackrel{?}{=} \psi'_{k,j} \cdot K'_{k,3j-1} + K'_{k,3j}$ . If the verification fails for any  $j$ ,  $\mathbf{R}$  concludes that  $B_1$  is faulty and stops. Otherwise,  $\mathbf{R}$  recovers  $\forall j, K'_{k-1,j}$  as  $\psi'_{k,j} + K_{k,3j-2}^{-1}$ .

---

$\dagger$ We denote  $\bar{1} = 2$  and vice-versa.

If at the end of the loop,  $\mathbf{R}$  has recovered  $K'_{1,1}, K'_{1,2}, K'_{1,3}$  then  $\mathbf{R}$  verifies whether  $\phi'_{1,1} \stackrel{?}{=} \psi'_{1,1} \cdot K'_{1,2} + K'_{1,3}$ . If the verification passes,  $\mathbf{R}$  recovers  $m_1 = \psi'_{1,1} + K'_{1,1}^{-1}$  as the message.

This completes the description of  $\Pi_1$ . The protocols  $\Pi_1$  and  $\Pi_2$  are run on network  $\mathcal{N}$ .  $\mathbf{R}$  takes one of the following actions based on the outcomes of these protocols: (a) If  $\mathbf{R}$  detects that  $B_i$  is corrupt in  $\Pi_i$ , it outputs whatever message it recovered from  $\Pi_{\bar{i}}$ . (b) If  $\mathbf{R}$  recovers messages from each of the  $\Pi_i$ 's and both the messages are same, it outputs that message. (c) If messages recovered through  $\Pi_1$  and  $\Pi_2$  are different, it outputs  $\perp$ . This completes  $\Pi$ . In the following lemma we prove that this is a  $(B, \frac{1}{|\mathbb{F}|})$ -URMT<sub>LV</sub> protocol.

**LEMMA 13.**  $\Pi$ , as constructed above, is a  $(B, \frac{1}{|\mathbb{F}|})$ -URMT<sub>LV</sub> protocol.

**PROOF.** We analyse the protocol case wise: (a) For some  $i$ ,  $\mathbf{R}$  concludes through  $\Pi_i$  that  $B_i$  is faulty, and outputs whatever it recovers from  $\Pi_{\bar{i}}$ . For each  $i$ , none of the nodes in  $B_{\bar{i}}$  participate in the protocol  $\Pi_i$ . Hence, if some verification fails during  $\Pi_i$ ,  $B_i$  has to be faulty, and  $\Pi_{\bar{i}}$  should recover the correct message  $m$ . (b) For each  $i \in \{1, 2\}$ , all verifications in  $\Pi_i$  pass. Case (i)  $m_i = m_{\bar{i}}$ ,  $\mathbf{R}$  outputs  $m_i$ . Since one of  $m_i$  or  $m_{\bar{i}}$  has to be same as  $m$ ,  $\mathbf{R}$ 's output is correct. This happens with  $\leq \frac{1}{|\mathbb{F}|}$  probability.  $\blacksquare$

### 3.2 Necessity

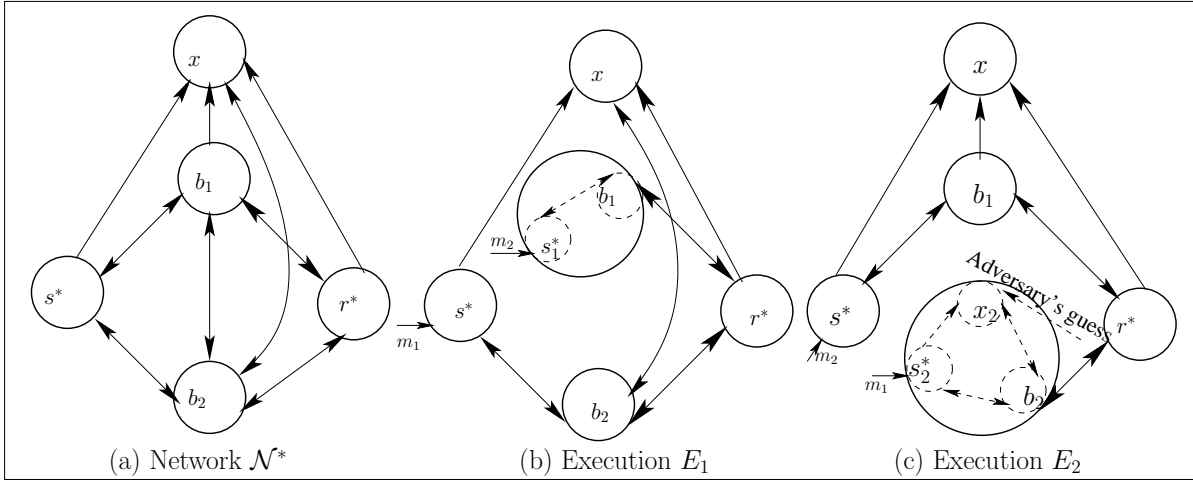


Figure 2: (a) The directed network  $\mathcal{N}^*$  (b) Adversary strategy when  $b_1$  is faulty (c) Adversary strategy when  $b_2$  is faulty. To complete the simulation of  $x$ , adversary feeds  $x_2$  with random elements. Execution  $E_2$  is one in which 'Adversary's guess' is what  $r^*$  sends to  $x$  in  $E_1$  and coin tosses with respect to all players (described in the Lemma 15) are such that  $r^*$  is bound to output  $m_1$  in  $E_2$ .

Let  $\mathcal{N}$  be a network that does not satisfy the conditions of Theorem 12. We show that in such a network  $(\{B_1, B_2\}, \delta)$ -URMT<sub>LV</sub> from  $\mathbf{S}$  to  $\mathbf{R}$  is impossible. Without loss of generality,



let us assume that the two sets comprising the adversary structure are disjoint<sup>†</sup>. Let the path  $q_1$ <sup>§</sup> be not present between  $\mathbf{S}$  and  $\mathbf{R}$  in  $\mathcal{N}$ . Hence, every weak path between  $\mathbf{S}$  and  $\mathbf{R}$  avoiding nodes in  $B_1 \cup B_2$  has at least one node  $w$  such that every strong path from  $w$  to  $\mathbf{R}$  passes through  $B_2$ .

We first consider the simple network  $\mathcal{N}^* = (V^*, \mathcal{E}^*)$  shown in Figure 2(a) consisting of five nodes  $s^*, r^*, b_1, b_2$  and  $x$  where  $s^*$  is the sender and  $r^*$  is the receiver and show that  $(\{\{b_1\}, \{b_2\}\}, \delta)$ -URMT<sub>LV</sub> from  $s^*$  to  $r^*$  is impossible in Lemma 14. We then show that the digraph  $\mathcal{N}$  can be partitioned into disjoint sets whose connectivity properties are similar to the connectivity between nodes of digraph  $\mathcal{N}^*$  in Lemma 16. We then prove in Lemma 17 that if  $(\{B_1, B_2\}, \delta)$ -URMT<sub>LV</sub> from  $\mathbf{S}$  to  $\mathbf{R}$  is possible in  $\mathcal{N}$  then  $(\{\{b_1\}, \{b_2\}\}, \delta)$ -URMT<sub>LV</sub> from  $s^*$  to  $r^*$  is also possible in  $\mathcal{N}^*$ , which is a contradiction. Hence, the conditions mentioned in Theorem 12 are necessary.

**LEMMA 14.** *In the synchronous network  $\mathcal{N}^*$ , shown in Figure 2(a),  $(\{\{b_1\}, \{b_2\}\}, \delta)$ -URMT<sub>LV</sub> ( $\delta < 1/2$ ) from  $s^*$  to  $r^*$  is impossible.*

**PROOF.** We assume that a protocol  $\Pi^*$  exists in  $\mathcal{N}^*$  which is a  $(\{\{b_1\}, \{b_2\}\}, \delta)$ -URMT<sub>LV</sub> protocol. We describe an adversary strategy  $\mathcal{S}$  to fail any protocol  $\Pi^*$  and prove it's correctness in the following lemma. Adversary chooses any two messages  $m_1, m_2 \in \mathbb{F}$ ,  $m_1 \neq m_2$ . When  $s^*$  intends to send  $m_i$ , the adversary corrupts the set  $b_i$  and snaps all communication with the nodes:  $b_i, x$  and  $s^*$ .

When adversary corrupts  $b_1$ , it simulates a local copy of  $s^*$  on input  $m_2$ , say  $s_1^*$ . At the beginning of each round,  $b_1$  receives messages from  $r^*$  and from the simulated  $s_1^*$ , does local computation and sends out messages to  $r^*$  and  $s_1^*$ . During the same round,  $s_1^*$  receives messages from  $b_1$ , its state is updated and messages are sent out for the next round.

When adversary corrupts  $b_2$ , it simulates a local copy of  $s^*$  on input  $m_1$  and a local copy of  $x$ , say  $s_2^*$  and  $x_2$  respectively. It handles the simulated  $s_2^*$  and  $x_2$  locally in the same manner as it handled the simulated  $s_1^*$  when  $b_1$  was corrupted. For simulation of  $x$  to happen,  $x_2$  is to be fed with some input on behalf of  $r^*$ , since an edge  $(r^*, x) \in \mathcal{E}^*$ . Adversary guesses the messages sent along this edge and feeds those to  $x_2$  round by round until the protocol terminates. Note that the node  $x$  has no strong path to  $r^*$  and hence does not have any influence on  $\mathbf{R}$ 's output.

For the sake of clarity, a pictorial view of the adversary strategy is shown in the Figure 2(b), (c). We prove that the above strategy fails every protocol  $\Pi^*$  in the following Lemma.

■

**LEMMA 15.** *With the adversary strategy  $\mathcal{S}$ , no protocol  $\Pi^*$  is a  $\{\{b_1\}, \{b_2\}\}$ -URMT<sub>LV</sub> protocol.*

**PROOF.** Before proceeding to the proof we introduce the following notations w.r.t. to an execution  $E_i$  of the protocol  $\Pi^*$ : (a) The vector  $\vec{C}_i = (c_{s^*}^i, c_{r^*}^i, c_{b_1}^i, c_{b_2}^i, c_x^i)$  which denotes the coin tosses input to nodes, where  $c_n^i$  denotes the coin tosses of node  $n$ . (b) The view of a node

<sup>†</sup>In case  $B_1 \cap B_2 \neq \emptyset$ , adversary strategy to fail any protocol in  $\mathcal{N}$  includes *fail-stopping* the nodes in the intersection.

<sup>§</sup>The case when the path  $q_2$  is not present from  $\mathbf{S}$  to  $\mathbf{R}$  can be handled analogously.

$n$ ,  $view_n(E_i)$ , which comprises of the internal coin tosses  $c_n^i$  of node  $n$  and the messages it receives during execution  $E_i$ . We now consider the following two executions:

1. There exists an execution  $E_1$  of  $\Pi^*$  such that  $s^*$  chooses to send  $m_1$  and  $r^*$  outputs  $m_1$  when the random tosses used by  $s_1^*$  are denoted by a string  $r$ , for otherwise  $\Pi^*$  won't be a  $\{\{b_1\}, \{b_2\}\}$ -URMT<sub>LV</sub> protocol.
2. Execution  $E_2$ :  $s^*$  chooses to send  $m_2$ . Coin tosses  $\vec{C}_2$  of nodes are such that  $c_{b_1}^2 = c_{b_1}^1$ ,  $c_{r^*}^2 = c_{r^*}^1$  and  $c_{s^*}^2 = r$ . Coin tosses of  $s_2^*$  and  $x_2$  are  $c_{s^*}^1$  and  $c_x^1$  respectively. Messages fed to  $x_2$  by the adversary matches exactly the messages sent by  $r^*$  to  $x$  in  $E_1$ .

For the above mentioned executions  $E_1$  and  $E_2$ ,  $view_{r^*}(E_1) = view_{r^*}(E_2)$ . Hence  $r^*$  halts with output  $m_1$  in execution  $E_2$ , violating the condition of  $\Pi^*$  being a URMT<sub>LV</sub> protocol.  $\blacksquare$

**LEMMA 16.** *The set of nodes  $V$  in the network  $\mathcal{N}$  can be partitioned into 5 disjoint sets  $S^*, R^*, B'_1 \subseteq B_1, B_2$  and  $X'$  such that  $\mathbf{S} \in S^*, \mathbf{R} \in R^*$  and an edge exists from a node in  $\vec{L}[i]$  to a node in  $\vec{L}[j]$  only if  $(\vec{L}[i], \vec{L}[j]) \in \mathcal{E}^*$  where  $\vec{L} = [S^*, R^*, B'_1, B_2, X']$  and  $\vec{l} = [s^*, r^*, b_1, b_2, x]$  are two ordered lists,  $\vec{L}[i]$  (resp.  $\vec{L}[i]$ ) denotes the  $i^{\text{th}}$  element of the list  $\vec{L}$  (resp.  $\vec{l}$ ).*

**PROOF.** In the network  $\mathcal{N}$ , every weak path between  $\mathbf{S}$  and  $\mathbf{R}$  avoiding  $B_1 \cup B_2$  has at least one node  $w$  such that every strong path from  $w$  to  $\mathbf{R}$  passes through  $B_2$ .

We partition the non-faulty nodes  $H = V \setminus \{B_1 \cup B_2\}$  into 3 disjoint sets namely:  $R^*, S^*$  and  $X$  defined as follows.  $R^* = \{w \mid w \in H \text{ and } \exists \text{ a weak path } p \text{ between } w \text{ and } \mathbf{R} \text{ s.t all the nodes in } p \text{ have a strong path to } \mathbf{R} \text{ avoiding nodes in } B_2\}$ .  $S^* = \{w \mid w \in H \setminus R^* \text{ and } w \text{ has a strong path to } \mathbf{R} \text{ avoiding } B_2\}$ .  $X = H \setminus \{S^* \cup R^*\}$ . Clearly,  $\mathbf{R} \in R^*$  and  $\mathbf{S} \in S^*$ . Moreover, if any node  $w \in X$  has a strong path to  $\mathbf{R}$ , it passes through some node in  $B_2$ .

We now divide the set  $B_1$  into two disjoint sets namely:  $B'_1$  and  $B_1^X$ .  $B'_1 = \{u \mid u \in B_1 \text{ and } u \text{ has a strong path to } \mathbf{R} \text{ avoiding } B_2\}$ .  $B_1^X = B_1 \setminus B'_1$ . We consider two sets  $X$  and  $B_1^X$  together as a set  $X'$  i.e.  $X' = X \cup B_1^X$ .

It trivially follows from the definitions above that  $\nexists (u, v) \in \mathcal{E}$  such that  $u \in X'$  and  $v \in S^* \cup R^* \cup B'_1$ , otherwise there would be a path from a node in  $X'$  to  $\mathbf{R}$  avoiding  $B_2$ . Also, there cannot exist any directed edge between a node in  $S^*$  and a node in  $R^*$ . Note the only edges missing from  $\mathcal{N}^*$  are  $(x, s^*), (x, r^*), (x, b_1)$  and  $(s^*, r^*), (r^*, s^*)$ . Hence, proved.  $\blacksquare$

**LEMMA 17.** *In the directed synchronous network  $\mathcal{N} = (V, \mathcal{E}), (\{B_1, B_2\}, \delta)$ -URMT<sub>LV</sub> is possible from  $\mathbf{S}$  to  $\mathbf{R}$  only if  $(\{\{b_1\}, \{b_2\}\}, \delta)$ -URMT<sub>LV</sub> is possible from  $s^*$  to  $r^*$  in the network  $\mathcal{N}_1^*$ .*

**PROOF.** We show how a  $(\{B_1, B_2\}, \delta)$ -URMT<sub>LV</sub> protocol  $\mathcal{P}$  on  $\mathcal{N}$  can be simulated on  $\mathcal{N}^*$  to obtain a  $(\{\{b_1\}, \{b_2\}\}, \delta)$ -URMT<sub>LV</sub> protocol. We simulate a virtual network  $\mathcal{N}$  over  $\mathcal{N}^*$  such that  $\mathcal{P}$  when run over the virtual  $\mathcal{N}$  is a  $(\{\{b_1\}, \{b_2\}\}, \delta)$ -URMT<sub>LV</sub> protocol. Simulation runs as follows: Node  $\vec{l}[i]$  simulates all the nodes in the set  $\vec{L}[i]$ . Edges in  $\mathcal{N}$  lie in one of the following two categories: (i) Those amongst the nodes within a  $\vec{L}[i]$ . Such edges can be simulated by  $\vec{l}[i]$  internally. (ii) Edges from a node in  $\vec{L}[i]$  to a node in  $\vec{L}[j]$ . We proved in the previous lemma that, an edge exists from a node in  $\vec{L}[i]$  to a node in  $\vec{L}[j]$  only if there exists an edge  $(\vec{L}[i], \vec{L}[j]) \in \mathcal{E}^*$ . Hence, such an edge can be simulated on edge  $(\vec{l}[i], \vec{l}[j])$ . Now, protocol  $\mathcal{P}$  is run on the virtual  $\mathcal{N}$  which is a  $(\{\{b_1\}, \{b_2\}\})$ -URMT<sub>LV</sub> protocol from  $s^*$  to  $r^*$  in  $\mathcal{N}^*$ .  $\blacksquare$

## 4 Characterizing asynchronous networks for $(\mathbb{A}, \delta)$ -URMT<sub>MC</sub>

In the following theorem we show that in the case of asynchronous networks, the problem of characterizing networks for (the (im)possibility of)  $(\mathbb{A}, \delta)$ -URMT<sub>MC</sub> reduces to the problem of characterizing networks for  $(B, \delta)$ -URMT<sub>MC</sub>, where  $|B| = 2$ .

**THEOREM 18.** *In a directed asynchronous network  $\mathcal{N}$ ,  $(\mathbb{A}, \delta)$ -URMT<sub>MC</sub> protocol is possible if and only if for every adversary structure  $B \subseteq \mathbb{A}$  such that  $|B| = 2$ ,  $(B, \delta)$ -URMT<sub>MC</sub> protocol is possible.*

The proof takes an approach similar to the one taken in the proof of Theorem 10. However, since the network is asynchronous, the way we build a protocol tolerating larger sized adversary structure from protocols tolerating smaller sized ones changes.

**PROOF.** *Necessity:* Obvious. *Sufficiency:* The proof takes an approach similar to the one taken in the proof of Theorem 10. However, since the network is asynchronous, the way we build a protocol tolerating larger sized adversary structure from protocols tolerating smaller sized ones changes. Let  $f \in \mathbb{F}$  be any element **S** intends to send to **R**: Consider  $\mathcal{A}$  and its three subsets  $\mathcal{A}_1, \mathcal{A}_2$  and  $\mathcal{A}_3$  as described in Theorem 10. For  $i \in \{1, 2, 3\}$ , let  $\zeta_i$  be a  $(\mathcal{A}_i, \frac{\delta}{2})$ -URMT<sub>MC</sub> protocol which can be constructed easily by repeating  $(\mathcal{A}_i, \delta)$ -URMT<sub>MC</sub> sufficiently many times, keeping **S**'s input same as  $f$ , in order to amplify the probability of success. The protocol  $\eta$  which is an  $(\mathcal{A}, \delta)$ -URMT<sub>MC</sub> protocol (as proved in the following lemma) is constructed as follows:

- For each  $i \in \{1, 2, 3\}$ , sub-protocols  $\zeta_i$  are run in parallel on  $f$ .
- **R** waits until two of the three  $\zeta_i$  sub-protocols terminate with same output and outputs that as the message.

**LEMMA 19.** *For the directed synchronous network  $\mathcal{N}$ , the protocol  $\eta$  constructed above is a  $(\mathcal{A}, \delta)$ -URMT<sub>MC</sub> protocol.*

**PROOF.** Any set  $B \in \mathcal{A}$  is present in at least two subsets among  $\mathcal{A}_1, \mathcal{A}_2$  and  $\mathcal{A}_3$ ; say the two subsets are  $\mathcal{A}_2$  and  $\mathcal{A}_3$ . Hence the two sub-protocols  $\zeta_2$  and  $\zeta_3$  terminate with the correct output with at least  $1 - \frac{\delta}{2}$  probability each. As **R** waits until two of the three  $\zeta_i$  sub-protocols terminate with same output,  $\eta$  fails only if at least one of  $\zeta_2$  and  $\zeta_3$  terminates with an incorrect message or does not terminate at all. Since this happens with at most  $1 - (1 - \frac{\delta}{2})^2$  probability,  $\eta$  is an  $(\mathcal{A}, \delta - \frac{\delta^2}{4})$ -URMT<sub>MC</sub> i.e. an  $(\mathcal{A}, \delta)$ -URMT<sub>MC</sub> protocol. ■

Having reduced the problem of URMT<sub>MC</sub> in an asynchronous network tolerating an adversary structure to the problem of URMT<sub>MC</sub> tolerating all its 2-sized subsets, we now proceed to characterize directed asynchronous network in which URMT<sub>MC</sub> tolerating adversary structure  $B = \{B_1, B_2\}$  is possible (where  $B_1, B_2 \in \mathbb{A}$ ).

**THEOREM 20.** *In a directed asynchronous network  $\mathcal{N}$ ,  $(B, \delta)$ -URMT<sub>MC</sub> protocol is possible if and only if for each  $\alpha \in \{1, 2\}$ , there exists a weak path  $q_\alpha$  avoiding nodes in  $B_1 \cup B_2$  such that every node  $u$  along the path  $q_\alpha$  has a strong path to **R** avoiding all nodes in  $B_{\bar{\alpha}}$ . (Paths  $q_1, q_2$  need not be distinct.)*

We give the sufficiency and the necessity proofs in the following sub-sections.

## 4.1 Sufficiency

The protocol for the sufficiency proof of above theorem is constructed in a manner similar to the synchronous Las Vegas protocol in Section 3.1. However, there are some important differences.

For a directed asynchronous network  $\mathcal{N}$ , which satisfies the conditions given in Theorem 20, we show how to construct a protocol  $\Pi$  tolerating the adversary structure  $B = \{B_1, B_2\}$ . If either  $q_1$  or  $q_2$  is a strong path from  $\mathbf{S}$  to  $\mathbf{R}$ ,  $\mathbf{S}$  trivially sends  $m$  along that path and  $\mathbf{R}$  is bound to receive it. When this is not the case, we construct two sub-protocols  $\Pi_1$  and  $\Pi_2$ . For each  $i \in \{1, 2\}$ , protocol  $\Pi_i$  uses the honest weak path  $q_i$ . We give a construction of  $\Pi_1$ , and the construction of  $\Pi_2$  follows by symmetry. We represent weak path  $q_1$  as  $y_0, u_1, y_1, \dots, u_n, y_n, u_{n+1}$  (as done in Section 3.1).  $\Pi_1$  proceeds in the following steps:

1.  $\mathbf{S}$  sends  $m$  to  $u_1$  along  $q_1$ . For  $1 \leq k \leq n$ , node  $y_k$  chooses  $3^k$  random keys namely  $K_{k,1}, K_{k,2}, \dots, K_{k,3^k}$  and sends those to  $u_k$  and  $u_{k+1}$ .
2. Node  $u_1$  waits for  $m$  to arrive from  $\mathbf{S}$  and keys  $K_{1,1}, K_{1,2}, K_{1,3}$  to arrive from  $y_1$ . It calculates  $(\psi_{1,1}, \phi_{1,1}) = \chi(m; K_{1,1}, K_{1,2}, K_{1,3})$  and sends it to  $\mathbf{R}$  along a strong path avoiding  $B_2$ .

For  $1 < k \leq n$ ,  $u_k$  waits for  $3^{k-1}$  keys to arrive from  $y_{k-1}$  and  $3^k$  keys to arrive from  $y_k$ <sup>¶</sup>. It authenticates the keys received from  $y_{k-1}$  with the keys received from  $y_k$  and sends it to  $\mathbf{R}$  along a strong path avoiding  $B_2$ . Formally,  $u_k$  calculates  $\forall j \ 1 \leq j \leq 3^{k-1} \ (\psi_{k,j}, \phi_{k,j}) = \chi(K_{k-1,j}; K_{k,3j-2}, K_{k,3j-1}, K_{k,3j})$ .

3.  $\mathbf{R}$  waits for  $\{K'_{n,1}, K'_{n,2}, \dots, K'_{n,3^n}\}$  to arrive from  $y_n$ .  $\mathbf{R}$  runs the following loop:

for  $k$  in  $n$  to 2

$\mathbf{R}$  waits until it receives  $\forall j \ 1 \leq j \leq 3^{k-1}, (\psi'_{k,j}, \phi'_{k,j})$  from  $u_k$ <sup>†</sup>. If  $\mathbf{R}$  does receive, it verifies  $\forall j, \phi'_{k,j} \stackrel{?}{=} \psi'_{k,j} \cdot K'_{k,3j-1} + K'_{k,3j}$ . If the verification fails for any  $j$ ,  $\mathbf{R}$  concludes that  $B_1$  is faulty and stops. Otherwise,  $\mathbf{R}$  recovers  $K'_{k-1,j}$  as  $\psi'_{k,j} + K_{k,3j-2}^{-1}$ , for every  $j$ .

If at the end of the loop,  $\mathbf{R}$  has recovered  $K'_{1,1}, K'_{1,2}$  and  $K'_{1,3}$  then  $\mathbf{R}$  waits to receive  $(\psi'_{1,1}, \phi'_{1,1})$  and verifies if  $\phi'_{1,1} \stackrel{?}{=} \psi'_{1,1} \cdot K'_{1,2} + K'_{1,3}$ . If the verification passes,  $\mathbf{R}$  recovers  $m_1 = \psi'_{1,1} + K'_{1,1}^{-1}$  as the message.

This completes the description of  $\Pi_1$ . The protocols  $\Pi_1$  and  $\Pi_2$  are run in parallel in the asynchronous network  $\mathcal{N}$ .  $\mathbf{R}$  takes one of the following actions based on the outcomes of these protocols: (a) If for any  $i \in \{1, 2\}$ ,  $\Pi_i$  concludes that  $B_i$  is faulty,  $\mathbf{R}$  waits for  $\Pi_{\bar{i}}$  to terminate, and outputs  $m_{\bar{i}}$  as message. (b) If for any  $i \in \{1, 2\}$ ,  $\Pi_i$  halts with  $m_i$  as message,  $\mathbf{R}$  outputs that as message without waiting for the protocol  $\Pi_{\bar{i}}$  to terminate. Above is a  $(B, \frac{1}{|\mathbb{F}|})$ -URMT<sub>MC</sub> protocol as proved in the following lemma.

<sup>¶</sup>As the communication between  $u_i$ 's and  $y_i$ 's occurs along the honest weak path  $q_1$ , every  $u_i$  receives the keys (or message) eventually.

<sup>†</sup>As these messages are delivered along faulty paths, they may never arrive. However, since  $\Pi_1$  and  $\Pi_2$  are run in parallel (as mentioned in the sequel) and  $\mathbf{R}$  waits for only one of them to terminate, the protocol  $\Pi$  always terminates.

**LEMMA 21.**  $\Pi$ , as constructed above, is a  $(B, \frac{1}{|\mathbb{F}|})$ -URMT<sub>MC</sub> protocol.

**PROOF.** We analyze the protocol case wise: (a) For some  $i$ ,  $\Pi_i$  outputs that  $B_i$  is faulty, and  $\mathbf{R}$  outputs what it recovers from  $\Pi_{\bar{i}}$ . None of the nodes in  $B_i$  participate in the protocol  $\Pi_{\bar{i}}$ . Therefore, some verification fails during  $\Pi_i$  only if  $B_i$  is faulty. Hence,  $\Pi_{\bar{i}}$  is bound to terminate with  $m_{\bar{i}} = m$ . (b) For some  $i$ ,  $\Pi_i$  terminates successfully with output  $m_i$ . Probability that  $m_i \neq m$  is at most  $\frac{1}{|\mathbb{F}|}$ . Hence,  $\mathbf{R}$ 's output is correct with probability at least  $\frac{|\mathbb{F}|-1}{|\mathbb{F}|}$ . ■

## 4.2 Necessity

Let  $\mathcal{N}$  be a network that does not satisfy the condition of 20. We show that in such a network  $(\{B_1, B_2\}, \delta)$ -URMT<sub>MC</sub> from  $\mathbf{S}$  to  $\mathbf{R}$  is impossible. Without loss of generality, we assume that the sets  $B_1$  and  $B_2$  are disjoint and path  $q_1$  is not present between  $\mathbf{S}$  and  $\mathbf{R}$  in  $\mathcal{N}$  (Reasons for the assumptions are clearly stated in Section 3.2). Hence, every weak path between  $\mathbf{S}$  and  $\mathbf{R}$  avoiding  $B_1 \cup B_2$  has at least one node  $w$  such that every strong path from  $w$  to  $\mathbf{R}$  passes through  $B_2$ . We again consider the simple network  $\mathcal{N}^* = (V^*, \mathcal{E}^*)$  shown in Figure 2(a) consisting of five nodes  $s^*, r^*, b_1, b_2$  and  $x$  where  $s^*$  is the sender and  $r^*$  is the receiver. However, this time the edges between nodes are asynchronous. We show that  $(\{\{b_1\}, \{b_2\}\}, \delta)$ -URMT<sub>MC</sub> from  $s^*$  to  $r^*$  is impossible in  $\mathcal{N}^*$  in Lemma 22. We then need to show that the digraph  $\mathcal{N}$  can be partitioned into disjoint sets whose connectivity properties are similar to the connectivity between nodes of digraph  $\mathcal{N}^*$ , which we have already proved in Lemma 16. Now, if  $(\{B_1, B_2\}, \delta)$ -URMT<sub>MC</sub> from  $\mathbf{S}$  to  $\mathbf{R}$  is possible in  $\mathcal{N}$  then  $(\{\{b_1\}, \{b_2\}\})$ -URMT<sub>MC</sub> from  $s^*$  to  $r^*$  is possible in  $\mathcal{N}^*$ , which leads us to a contradiction (We need not prove this separately as the proof given in Lemma 17 works even when both  $\mathcal{N}$  and  $\mathcal{N}^*$  are asynchronous networks). Hence, the conditions mentioned in Theorem 20 are necessary.

**LEMMA 22.** In the asynchronous network  $\mathcal{N}^*$ , shown in figure 2(a),  $(\{\{b_1\}, \{b_2\}\}, \delta)$ -URMT<sub>MC</sub> ( $\delta < 1/2$ ) from  $s^*$  to  $r^*$  is impossible.

**PROOF.** We assume that a protocol  $\Pi^*$  exists in  $\mathcal{N}^*$  which is a  $(\{\{b_1\}, \{b_2\}\}, \delta)$ -URMT<sub>MC</sub> protocol from  $s^*$  to  $r^*$ . This implies that there exists a finite time instant  $T$  such that for every message  $m$ ,  $r^*$  outputs  $m$  before instant  $T$  in at least half of the executions in which  $s^*$  has chosen to send  $m$ . We now describe an adversary strategy to fail any such protocol  $\Pi^*$ . Adversary chooses any two messages  $m_1, m_2 \in \mathbb{F}, m_1 \neq m_2$  and ensures that the probability that  $r^*$  outputs  $m_1$  given  $s^*$  has sent  $m_1$  is  $< \frac{1}{2}$ . When  $s^*$  intends to send  $m_i$ , it corrupts the node  $b_i$ , for  $i \in \{1, 2\}$ . When  $b_2$  is corrupt, adversary simply fail-stops it. When  $b_1$  is corrupt, adversary does the following:

- Simulates a local copy of  $s^*$  on input  $m_2$ , say  $s_1^*$ .
- Snaps all it's communication with the nodes:  $s^*, x$  and  $b_2$ .
- Delays all the outgoing messages from  $b_2$  beyond the time instant  $T$ . Schedules events between the rest of the nodes (and simulated nodes) i.e.  $s_1^*, r^*, b_1$  and  $x$  as it schedules events between  $s^*, r^*, b_1$  and  $x$  respectively when  $s^*$ 's input is  $m_2$  (Note that in an asynchronous network, the adversary is additionally equipped with the ability to schedule messages).



We now show that for every execution in which  $s^*$ 's input is  $m_2$ , there is an execution in which  $s^*$ 's input is  $m_1$  such that the view at  $r^*$  is same in both the executions. Consider the following two executions:-

1. Execution  $E_1$ :  $s^*$ 's input is  $m_2$  and coin tosses of  $s$ ,  $b_1$  and  $r^*$  are  $c_1$ ,  $c_2$  and  $c_3$  respectively. Note that in this execution the coin tosses of  $x$  does not affect the output at  $r^*$  as there is no strong path from  $x$  to  $r^*$ .
2. Execution  $E_2$ :  $s^*$ 's input is  $m_1$  and coin tosses of  $s_1^*$ ,  $b_1$  and  $r^*$  are  $c_1$ ,  $c_2$  and  $c_3$  respectively. Note that in such an execution the coin tosses of  $s^*$  and  $x$  does not affect the output at  $r^*$ .

It follows that for every such  $E_1$  there is a corresponding  $E_2$  and  $View_{r^*}(E_1) = View_{r^*}(E_2)$ . This ensure, the probability that  $r^*$  outputs  $m_2$  given  $s^*$  has sent  $m_1$  is same as the probability that  $r^*$  outputs  $m_2$  given  $s^*$  has sent  $m_2$ , which is  $> \frac{1}{2}$ . Hence, a contradiction. ■

**COROLLARY 23.** *In a directed network  $\mathcal{N} = (V, \mathcal{E})$ , a synchronous  $(\mathbb{A}, \delta)$ -URMT<sub>LV</sub> protocol exists if and only if a protocol exists for asynchronous  $(\mathbb{A}, \delta)$ -URMT<sub>MC</sub>.*

PROOF. Follows from Theorem 10, 12 and 18, 20. ■

## 5 Characterizing asynchronous networks for $(\mathbb{A}, \delta)$ -URMT<sub>LV</sub>

In the following theorem we show that in the case of asynchronous networks, the problem of characterizing networks for (the (im)possibility of)  $(\mathbb{A}, \delta)$ -URMT<sub>LV</sub> reduces to the problem of characterizing networks for  $(B, \delta)$ -URMT<sub>LV</sub>, where  $|B| = 2$ .

**THEOREM 24.** *In a directed asynchronous network  $\mathcal{N}$ ,  $(\mathbb{A}, \delta)$ -URMT<sub>LV</sub> protocol is possible if and only if for every adversary structure  $B \subseteq \mathbb{A}$  such that  $|B| = 2$ ,  $(B, \delta)$ -URMT<sub>LV</sub> protocol is possible.*

PROOF. Similar to the proof of Theorem 18, hence omitted. ■

Having reduced the problem of URMT<sub>LV</sub> in an asynchronous network tolerating an adversary structure to the problem of URMT<sub>LV</sub> tolerating all its 2-sized subsets, we now proceed to characterize directed asynchronous networks in which URMT<sub>LV</sub> tolerating adversary structure  $B = \{B_1, B_2\}$  is possible (where  $B_1, B_2 \in \mathbb{A}$ ).

**THEOREM 25.** *In a directed asynchronous network  $\mathcal{N}$ ,  $(B, \delta)$ -URMT<sub>LV</sub> protocol is possible if and only if there exists a strong path from **S** to **R** avoiding nodes in  $B_1 \cup B_2$ .*

PROOF. *Sufficiency:* Let  $f$  be the field element **S** intends to send. Send  $f$  to **R** along the strong path avoiding nodes in  $B_1 \cup B_2$ . Since, the path does not contain any corrupt nodes,  $f$  is eventually received by **R**. ■

We give the necessity proof of the above theorem in the following sub-section.

### 5.1 Necessity

The proof in this section is along similar lines to the necessity proofs earlier. Let  $\mathcal{N}$  be a network that does not satisfy the condition mentioned in Theorem 25. We first consider the



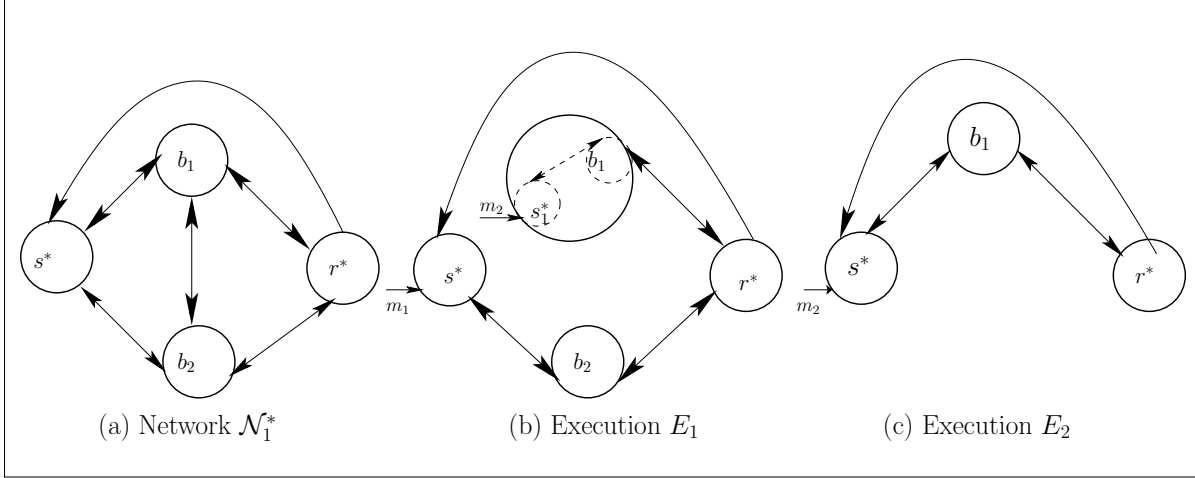


Figure 3: (a) The directed network  $\mathcal{N}_1^*$  (b) Adversary strategy when  $b_1$  is faulty (c) Adversary strategy when  $b_2$  is faulty.

simple asynchronous network  $\mathcal{N}_1^* = (V_1^*, \mathcal{E}_1^*)$  with  $V_1^* = \{s^*, r^*, b_1, b_2\}$  and  $\mathcal{E}_1^* = (V_1^* \times V_1^*) \setminus \{(s^*, r^*)\}$  as shown in Figure 3(a) and show that  $(\{\{b_1\}, \{b_2\}\})$ -URMT<sub>LV</sub> from  $s^*$  to  $r^*$  is impossible in Lemma 26. We then show that the digraph  $\mathcal{N}$  can be partitioned into four disjoint sets whose connectivity properties are similar to the connectivity between nodes of digraph  $\mathcal{N}_1^*$  in Lemma 27. Finally in Lemma 28, we show that if  $(\{B_1, B_2\}, \delta)$ -URMT<sub>LV</sub> from  $\mathbf{S}$  to  $\mathbf{R}$  is possible in  $\mathcal{N}$  then  $(\{\{b_1\}, \{b_2\}\})$ -URMT<sub>LV</sub> from  $s^*$  to  $r^*$  is possible in  $\mathcal{N}_1^*$ , which is a contradiction. Hence, the conditions mentioned in Theorem 25 are necessary.

**LEMMA 26.** *In the asynchronous network  $\mathcal{N}_1^*$ ,  $(\{\{b_1\}, \{b_2\}\}, \delta)$ -URMT<sub>LV</sub> ( $\delta < 1/2$ ) from  $s^*$  to  $r^*$  is impossible.*

**PROOF.** We assume that a protocol  $\Pi^*$  exists in  $\mathcal{N}_1^*$  which is a  $(\{\{b_1\}, \{b_2\}\}, \delta)$ -URMT<sub>LV</sub> protocol from  $s^*$  to  $r^*$ . This implies that there exists a finite time instant  $T$  such that for every message  $m$ ,  $r^*$  outputs  $m$  before instant  $T$  in at least half of the executions in which  $s^*$  has chosen to send  $m$ . We use  $\vec{C}_i$  and  $view_n(E_i)$  as defined in the proof of Lemma 14.

We now describe an adversary strategy to fail any protocol  $\Pi^*$ . Adversary chooses any two messages  $m_1, m_2 \in \mathbb{F}$ ,  $m_1 \neq m_2$ . When  $s^*$  intends to send  $m_i$ , the adversary corrupts the set  $\{b_i\}$ , for  $i \in \{1, 2\}$ . When  $b_2$  is corrupt, adversary fail-stops it. When  $b_1$  is corrupt, adversary does the following:-

- Simulates a local copy of  $s^*$  on input  $m_2$ , say  $s_1^*$ .
- Snaps all its communication with nodes  $s^*$  and  $b_2$ .
- Delays all outgoing messages from  $b_2$  beyond the time instant  $T$ . Schedules events between  $s_1^*$ ,  $b_1$ , and  $r^*$  as it does between  $s^*$ ,  $b_1$ , and  $r^*$  respectively when  $s^*$ 's input is  $m_2$ .

We now consider the following two executions of  $\Pi^*$  with the above mentioned adversary strategy.

1. Execution  $E_1$ :  $s^*$ 's input is  $m_1$  and  $r^*$  outputs  $m_1$ . (For  $\Pi^*$  to be a valid  $(\{\{b_1\}, \{b_2\}\})$ -URMT<sub>LV</sub> protocol, such an execution exists). Let the coin tosses of  $s_1^*$  be  $c$ .

2. Execution  $E_2$ :  $s^*$ 's input is  $m_2$ . Coin tosses  $\vec{C}_2$  of nodes are such that  $c_{s^*}^2 = c$ ,  $c_{r^*}^2 = c_{r^*}^1$ ,  $c_{b_1}^2 = c_{b_1}^1$ .

Above mentioned adversary strategy ensures that  $view_{r^*}(E_2) = view_{r^*}(E_1)$ . Therefore  $r^*$  halts with output  $m_1$  in  $E_2$  which is a contradiction on the existence of  $\Pi^*$  since Las Vegas protocols do not allow incorrect output.  $\blacksquare$

We now consider network  $\mathcal{N} = (V, \mathcal{E})$  which does not satisfy the conditions of Theorem 25.

**LEMMA 27.** *The set of nodes  $V$  in the network  $\mathcal{N}$  can be partitioned into 4 disjoint sets  $S^*, B_1, B_2$  and  $R^*$  such that  $\mathbf{S} \in S^*$ ,  $\mathbf{R} \in R^*$  and an edge exists from a node in  $\vec{L}[i]$  to a node in  $\vec{L}[j]$  only if  $(\vec{l}[i], \vec{l}[j]) \in \mathcal{E}_1^*$  where  $\vec{L} = [S^*, B_1, B_2, R^*]$  and  $\vec{l} = [s^*, b_1, b_2, r^*]$  are two ordered lists,  $\vec{l}[i]$  (resp.  $\vec{L}[i]$ ) denotes the  $i^{\text{th}}$  element of the list  $\vec{l}$  (resp.  $\vec{L}$ ).*

**PROOF.** We partition the non-faulty nodes in  $H = V \setminus \{B_1 \cup B_2\}$  into 2 disjoint sets  $S^*$  and  $R^*$ . Let  $R^*$  denote the set of all nodes in  $H$  having a strong path to  $\mathbf{R}$  avoiding nodes in  $B_1 \cup B_2$ . Let  $S^* = V \setminus \{R^* \cup B_1 \cup B_2\}$ . There does not exist a directed edge from a node in  $S^*$  to a node in  $R^*$  by the definition of  $R^*$ . Since,  $\mathcal{E}_1^* = (V_1^* \times V_1^*) \setminus \{(s^*, r^*)\}$ , an edge exists from a node in  $\vec{L}[i]$  to a node in  $\vec{L}[j]$  only if there exists an edge  $(\vec{l}[i], \vec{l}[j]) \in \mathcal{E}_1^*$ .  $\blacksquare$

**LEMMA 28.** *In the directed asynchronous network  $\mathcal{N} = (V, \mathcal{E})$ ,  $(\{B_1, B_2\}, \delta)$ -URMT<sub>LV</sub> is possible from  $\mathbf{S}$  to  $\mathbf{R}$  only if  $(\{\{b_1\}, \{b_2\}\}, \delta)$ -URMT<sub>LV</sub> is possible from  $s^*$  to  $r^*$  in the network  $\mathcal{N}_1^*$ .*

**PROOF.** Similar to the proof of Lemma 17, hence omitted.  $\blacksquare$

**THEOREM 29.** *In a directed synchronous (or asynchronous) network  $\mathcal{N} = (V, \mathcal{E})$ ,  $\mathbb{A}$ -PRMT from  $\mathbf{S}$  to  $\mathbf{R}$  is possible if and only if for all  $B_1, B_2 \in \mathbb{A}$  there exists a strong path from  $\mathbf{S}$  to  $\mathbf{R}$  avoiding nodes in  $B_1 \cup B_2$ .*

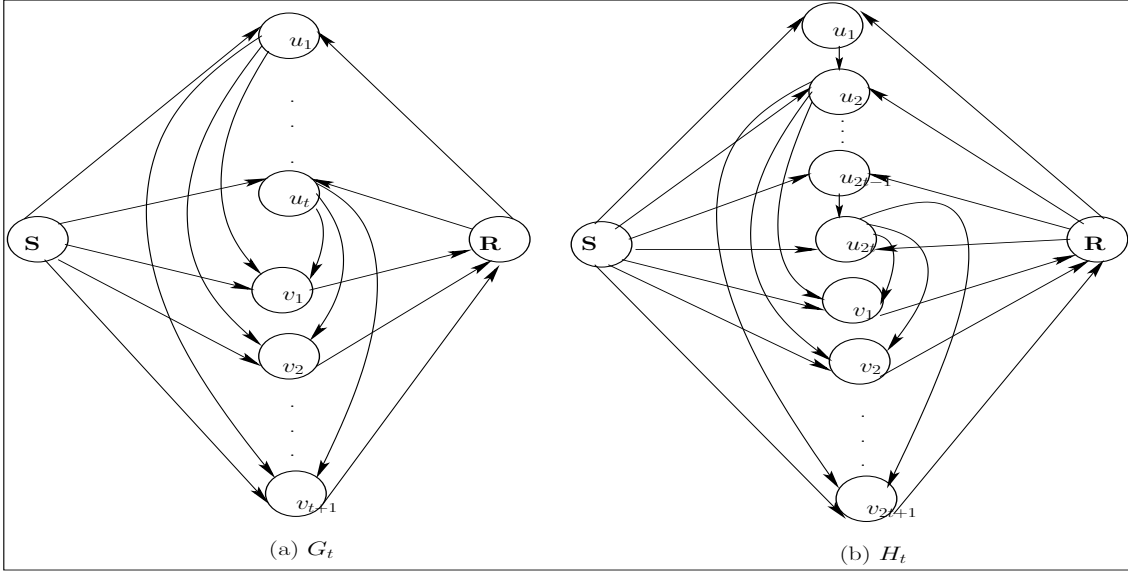
**PROOF.** Follows from [3].  $\blacksquare$

**COROLLARY 30.** *In a directed network  $\mathcal{N} = (V, \mathcal{E})$ , an asynchronous  $(\mathbb{A}, \delta)$ -URMT<sub>LV</sub> protocol exists if and only if a protocol exists for synchronous (or asynchronous)  $\mathbb{A}$ -PRMT.*

**PROOF.** Follows from Theorem 24, 25 and 29.  $\blacksquare$

## 6 Critical edges

In [8], Bhavani et. al. proposed a family of graphs ( $t > 0$ ),  $G_t = (V, \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3)$  where  $V = \{\mathbf{S}, v_1, \dots, v_{t+1}, u_1, \dots, u_t, \mathbf{R}\}$  and  $\mathcal{E}_1 = \bigcup_{i=1}^{t+1} \{(\mathbf{S}, v_i), (v_i, \mathbf{R})\}$ ;  $\mathcal{E}_2 = \bigcup_{i=1}^t \{(\mathbf{S}, u_i), (\mathbf{R}, u_i)\}$ ;  $\mathcal{E}_3 = \bigcup_{i=1}^t \{(u_i, v_1), \dots, (u_i, v_{t+1})\}$  as shown in Figure 4(a) and claimed that  $G_t$  has  $\Omega(n^2)$  critical edges w.r.t synchronous  $(t, \delta)$ -URMT<sub>MC</sub>. In the following theorem, we prove that this is not the case.

Figure 4: (a) Graph  $G_t$ , (b) Graph  $H_t$ 

**THEOREM 31.**  $G_t$  has only  $\Theta(n)$  critical edges w.r.t synchronous  $(t, \delta)$ -URMT<sub>MC</sub>.

PROOF. Consider the subgraph  $G'_t$  of  $G_t$  given by  $G'_t = (V, \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}'_3)$ , where  $\mathcal{E}'_3 = \bigcup_{i=1}^t \{u_i, v_i\}$ . According to [8], for  $(t, \delta)$ -URMT<sub>MC</sub> to be impossible in  $G'_t$ , there exists  $B_1, B_2$  such that  $|B_1|, |B_2| \leq t$  and every weak path from  $\mathbf{S}$  to  $\mathbf{R}$  avoiding nodes in  $B_1 \cup B_2$  must have at least one node  $x$  such that every strong path from  $x$  to  $\mathbf{R}$  passes through nodes in  $B_1$  and in  $B_2$ . We first show that no such sets  $B_1, B_2$  exists for  $G'_t$ .

If there exists a strong path from  $\mathbf{S}$  to  $\mathbf{R}$  avoiding nodes in  $B_1 \cup B_2$ , URMT<sub>MC</sub> is trivially possible. Hence,  $\forall i \ 1 \leq i \leq t+1, v_i \in B_1$  or  $v_i \in B_2$ . As  $|B_1| + |B_2| \leq 2t$ , at least one  $u_i$  has to be honest. As this leaves an honest weak path from  $\mathbf{S}$  to  $\mathbf{R}$  i.e.  $\mathbf{S} \rightarrow u_i \leftarrow \mathbf{R}$  with  $u_i$  having a strong path to  $\mathbf{R}$  via  $v_i$ . For the impossibility of synchronous  $(t, \delta)$ -URMT<sub>MC</sub>, node  $v_i$  must belong to both  $B_1$  and  $B_2$ . This would imply that another  $u_{i'}$  ( $i' \neq i$ ) is honest and hence  $v_{i'}$  must belong to both  $B_1$  and  $B_2$ . Repeating the inductive arguing for another  $t-2$  times, we can show that  $B_1 = B_2 = \{v_{\alpha_1}, v_{\alpha_2}, \dots, v_{\alpha_t}\}$  for some  $\{\alpha_1, \alpha_2, \dots, \alpha_t\} \subset \{1, 2, 3, \dots, t+1\}$ . But this leaves a strong honest path from  $\mathbf{S}$  to  $\mathbf{R}$ . Hence, no  $B_1, B_2$  exists such that  $(B, \delta)$ -URMT<sub>MC</sub> is impossible in  $G'_t$  i.e.  $(t, \delta)$ -URMT<sub>MC</sub> is possible in  $G'_t$ .

Since  $G'_t$  has  $O(n)$  edges, this proves an upper bound of  $O(n)$  on the number of critical edges in  $G_t$ . Hence, the claim in [8] that  $G_t$  has  $\Omega(n^2)$  critical edges is wrong. Moreover, since deleting any one edge  $(\mathbf{S}, v_i)$  in  $G_t$  leaves only  $2t$  disjoint weak paths between  $\mathbf{S}$  and  $\mathbf{R}$ ,  $G_t$  has  $\Omega(n)$  critical edges. It therefore follows that  $G_t$  has  $\Theta(n)$  critical edges.  $\blacksquare$

We now propose a family of graphs with  $\Omega(n^2)$  critical edges. For all  $t > 0$ , consider  $H_t = (V^1, \bigcup_{i=1}^t \mathcal{E}_i^1)$  with  $V^1 = \{\mathbf{S}, v_1, \dots, v_{2t+1}, u_1, \dots, u_{2t}, \mathbf{R}\}$  and  $\mathcal{E}_1^1 = \bigcup_{i=1}^{2t+1} \{(\mathbf{S}, v_i), (v_i, \mathbf{R})\}$ ;  $\mathcal{E}_2^1 = \bigcup_{i=1}^{2t} \{(\mathbf{S}, u_i), (\mathbf{R}, u_i)\}$ ;  $\mathcal{E}_3^1 = \bigcup_{i=1}^t \{(u_{2i-1}, u_{2i})\}$ ;  $\mathcal{E}_4^1 = \bigcup_{i=1}^t \{(u_{2i}, v_1), \dots, (u_{2i}, v_{2t+1})\}$  as shown in Figure 4(b). Here, number of nodes in graph  $H_t$  is  $n = 4t + 3$ .

**THEOREM 32.**  $H_t$  has  $\Omega(n^2)$  critical edges w.r.t synchronous  $(2t, \delta)$ -URMT<sub>MC</sub>.

PROOF.  $(2t, \delta)$ -URMT<sub>MC</sub> is possible in  $H_t$  (Follows from [8]). Suppose we delete an edge  $e = (u_{2i}, v_j) \in \mathcal{E}_4^1$ . Consider  $B_1 = \bigcup_{k=1}^{2t} \{u_k\} \cup \{v_j\} - \{u_{2i-1}\}$ ,  $B_2 = \bigcup_{k=1}^{2t+1} \{v_k\} - \{v_j\}$ . Only honest weak path left is  $\mathbf{S} \rightarrow u_{2i-1} \leftarrow \mathbf{R}$ . Every strong path from  $u_{2i-1}$  to  $\mathbf{R}$  passes through both  $B_1$  and  $B_2$ . This renders  $(\{B_1, B_2\}, \delta)$ -URMT<sub>MC</sub> impossible, hence  $(2t, \delta)$ -URMT<sub>MC</sub> is impossible. Hence,  $H_t$  has  $\Omega(|\mathcal{E}_3^1|)$  or  $\Omega(n^2)$  critical edges. ■

We now give a family of digraphs with  $\Omega(n^2)$  critical edges w.r.t asynchronous  $(t, \delta)$ -URMT<sub>MC</sub> and  $(t, \delta)$ -URMT<sub>LV</sub>.

**THEOREM 33.**  $G_t$  has  $\Omega(n^2)$  critical edges w.r.t asynchronous  $(t, \delta)$ -URMT<sub>MC</sub> and synchronous  $(t, \delta)$ -URMT<sub>LV</sub>.

PROOF.  $(t, \delta)$ -URMT<sub>MC</sub> is possible in asynchronous network  $G_t$  (Follows from Theorem 18, 20). Suppose we delete some edge  $e = (u_i, v_j) \in \mathcal{E}_3$ . Consider  $B_1 = \bigcup_{k=1}^t \{u_k\} \cup \{v_j\} - \{u_i\}$ ,  $B_2 = \bigcup_{k=1}^{t+1} \{v_k\} - \{v_j\}$ . Only honest weak path left is  $\mathbf{S} \rightarrow u_i \leftarrow \mathbf{R}$ . All the strong paths from  $u_i$  to  $\mathbf{R}$  pass through  $B_2$ . This renders  $(\{B_1, B_2\}, \delta)$ -URMT<sub>MC</sub> impossible, hence  $(t, \delta)$ -URMT<sub>MC</sub> is impossible. Hence,  $\Omega(|\mathcal{E}_3|)$  or  $\Omega(n^2)$  critical edges. Since, asynchronous  $(t, \delta)$ -URMT<sub>MC</sub> is possible if and only if synchronous  $(t, \delta)$ -URMT<sub>LV</sub> is possible (Follows from Theorem 23), the result proved above holds for the synchronous  $(t, \delta)$ -URMT<sub>LV</sub> case too. ■

## 7 Discussion and Open Problems

Note that in this work we focus on the characterization of networks for possibility and impossibility of protocols. The protocols we gave in the sufficiency proofs have communication complexity exponential in the number of blocked nodes involved. Nonetheless, if the number of blocked nodes is small, we have the following interesting use-case which exploits the matching connectivity requirements of synchronous Las Vegas and asynchronous Monte Carlo URMT. Suppose a hardware vendor has to design a network such that Monte Carlo URMT between given two nodes is possible even when the network is asynchronous and deploy hardware for it. Testing of asynchronous networks is a tedious and time consuming job. So, he may rather test if the hardware achieves Las Vegas URMT when the network is synchronous, and be rest assured.

We briefly discuss a few related open problems: (a) There are real life networks where only directed hypergraphs can appropriately abstract the underlying network. Hence, it is useful to study the problem of URMT in directed hypergraphs under various timing models, fault models, etc. It would be interesting to see if the equivalence shown above for directed graphs in corollary 23, 30 extend to the case of directed hypergraphs as well. (b) Given a network  $\mathcal{N}$ , a sender  $\mathbf{S}$ , a receiver  $\mathbf{R}$  and an adversary  $\mathcal{A}$ , the decision problem of finding out whether URMT<sub>LV</sub> is not possible is trivially in the complexity class NP. It is non-trivial to prove this decision problem to be certain complexity class hard. (c) As we focused on characterizing directed networks in which URMT is possible in this paper, our protocols are inefficient in the worst case. It remains open to give efficient protocols or establish lower bounds. We strongly believe that the latter is the case. (d) In this paper we considered the

problem of reliable unicast wherein there is only one receiver. In general, one may consider the problem of *reliable multicast* wherein there are multiple recipients for the message the sender sends. The characterization of networks for the (im)possibility of reliable multicast is non-trivial because the problem definition demands all the receivers to agree on the same message when the sender is faulty which is trivial when there is only one receiver i.e. the case of unicast but non-trivial when number of receivers are more than 1.

## References

- [1] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 52–61, New York, NY, USA, 1993. ACM.
- [2] Y. Desmedt and Y. Wang. Perfectly Secure Message Transmission Revisited. In *Proceedings of Advances in Cryptology EUROCRYPT '02*, volume 2332 of *Lecture Notes in Computer Science (LNCS)*, pages 502–517. Springer-Verlag, 2002.
- [3] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *J. ACM*, 40(1):17–47, 1993.
- [4] M. Franklin and R. N. Wright. Secure Communication in Minimal Connectivity Models. In *Proceedings of Advances in Cryptology EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science (LNCS)*, pages 346–360. Springer-Verlag, 1998.
- [5] M. Hirt and U. Maurer. Player Simulation and General Adversary Structures in Perfect Multiparty Computation. *Journal of Cryptology*, 13(1):31–60, April 2000.
- [6] Abhinav Mehta, Shashank Agrawal, and Kannan Srinathan. Brief announcement: Synchronous Las Vegas URMT Iff Asynchronous Monte Carlo URMT. In Nancy A. Lynch and Alexander A. Shvartsman, editors, *DISC*, volume 6343 of *Lecture Notes in Computer Science*, pages 201–203. Springer, 2010.
- [7] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 73–85, New York, NY, USA, 1989. ACM.
- [8] Bhavani Shankar, Prasant Gopal, Kannan Srinathan, and C. Pandu Rangan. Unconditionally reliable message transmission in directed networks. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1048–1055, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.
- [9] Kannan Srinathan and C. Pandu Rangan. Possibility and complexity of probabilistic reliable communications in directed networks. In *Proceedings of 25th ACM Symposium on Principles of Distributed Computing (PODC'06)*, 2006.