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

Abhinav Mehta, Shashank Agrawal, and Kannan Srinathan

Center for Security, Theory and Algorithmic Research (C-STAR),
International Institute of Information Technology, Hyderabad, 500032, India.
{abhinav_mehta@research.,sagrawal@research.,srinathan@}iiit.ac.in

Abstract. We consider the distributed setting where a subset of nodes are under the control of a malicious adversary with unbounded computational power. The problem of simulating a directed reliable channel from a sender **S** to a receiver **R** in the absence of a true physical point to point channel is fundamental to the area of Distributed Computing and is popularly known as 'Unconditionally Reliable Message Transmission' (URMT).

In this work, we distinguish between two variants of URMT - *Monte Carlo* and *Las Vegas*. The former allows **R** to output an incorrect message with negligibly small probability, whereas the latter only allows **R** to abort the protocol with a small probability, but never to output an incorrect message.

We establish a novel hierarchy with respect to the connectivity requirements for URMT protocols to be possible over directed networks, under two extremes w.r.t. timing model: *synchronous* and *asynchronous*. We show that the minimum connectivity requirements for the existence of Las Vegas URMT protocols over synchronous networks is same as that of Monte Carlo URMT protocols over asynchronous networks - a surprising equivalence between two very different models. Furthermore, the higher connectivity requirements for Las Vegas URMT over asynchronous networks match exactly with that of zero-error (perfect) protocols over (a)synchronous networks.

We also show that for the 'easier' randomized variant (ones having less minimum connectivity requirements than perfect ones) the number of *critical edges* are higher than that of the perfect protocols, in the worst case. Hence, establishing an interesting interplay for the case of URMT.

1 Introduction

Most of the distributed computing protocols assume that every pair of participating nodes share a reliable channel, which is usually not true in practice. In the Unconditionally Reliable Message Transmission (URMT) problem, two non-faulty players, the sender **S** and the receiver **R**, are part of a communication network modelled as a directed graph over n players/nodes influenced by an unbounded adversary that may corrupt some subset of these n players/nodes. **S** has a message that it wishes to send to **R**; the challenge is to design a protocol such that **R** correctly obtains **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 **R** outputs the sender's message with high probability and may produce an incorrect output with small probability; in the latter, **R** outputs the sender's message with high probability and may abort the protocol with small probability, but in no case does the receiver terminate with an incorrect output. While Monte Carlo URMT has been studied in [13, 14], 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, over synchronous digraphs, achieving Las Vegas URMT is harder (and again it indeed requires more network connectivity) than achieving Monte Carlo URMT. 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 turn out to be the same as those for the existence of (a)synchronous perfect protocols.

We further improve our insights in the problem by studying the sparseness of digraphs which permit URMT. 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). While it is known that for perfect protocols the number of critical edges is always $O(n)$, it turns out that for the "easier" randomized protocols there exists a family of digraphs with $\Omega(n^2)$ critical edges! We remark that an earlier attempt in [13] 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

In [6], Dolev et al. initiate the study of message transmission protocols which provide both *perfect* secrecy and *perfect* resiliency by abstracting the network as a collection of n channels (corresponding to vertex-disjoint paths) between two synchronized non-faulty processors **S** and **R**. Franklin et al. [8] show that the connectivity requirements for Dolev et al.'s problem stay the same even when privacy is not required and there is a fairly large probability of failure of reliability (this is the general problem of URMT we have described above). Adopting Dolev et al.'s network abstraction, Ashish et al. [4] study several variants of message transmission in *asynchronous* networks.

While Dolev et al.'s work assumed all the channels between **S** and **R** to be either *1-way* (allowing information to flow from **S** to **R** only) or *2-way* (allowing information to flow in both directions), which corresponds naturally with undirected networks, in [5] Desmedt and Wang argue that a better way to model directed networks would be to have some channels in forward direction (from **S** to **R**) and some in backward direction. Several results have been derived in this model, see [11, 10, 15].

In [14], Srinathan and Rangan consider a more general setting, 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 allowed to perform computations on messages received, instead of merely forwarding these messages to other nodes. They provide the minimum connectivity required in a synchronous network for a Monte Carlo URMT[1] protocol tolerating a *mixed* adversary[2] to exist. In [13], Bhavani et al. obtain a much simpler connectivity requirement for the particular case of Byzantine adversary.

To the best of our knowledge, asynchronous networks have not been studied in this model.

1.2 Organization of paper

In the following section, we describe the network, fault and timing model we will be working with; provide definitions of various URMT problems; describe the two kinds of paths that exist in a network and their role in the design of protocols; and discuss a message authentication scheme that we will use extensively in designing our protocols.

In Sections 3, 4 and 5, we characterize directed networks for the possibility (and impossibility) of (i) Las Vegas URMT (URMT$_{LV}$) in synchronous networks, (ii) Monte Carlo URMT (URMT$_{MC}$) in asynchronous networks, and (iii) URMT$_{LV}$ in asynchronous networks respectively, tolerating a non-threshold Byzantine adversary. It is hard and non-intuitive to deal with an arbitrary sized adversary structure, so we take the following two step approach in each of these

[1] The problem of PRC is defined in this paper in exactly the same way as we define Monte Carlo URMT here (the adversary model is different though).

[2] A combination of Byzantine and fail-stop faults is considered in this paper.

sections: We first show that in order to tolerate an adversary structure $\mathbb{A}$ it is sufficient to tolerate all its two sized subsets. [3] We then give a necessary and sufficient condition on the connectivity of directed networks so that a protocol for the respective URMT variant tolerating a two-sized adversary structure exists.

In Section 6, we study the concept of *critical edges*. We first show that the family of digraphs over n nodes proposed in [13] for synchronous URMT$_{MC}$ claimed to have $\Omega(n^2)$ critical edges has only $O(n)$ critical edges. Then we present another family which has indeed $\Omega(n^2)$ critical edges. Further in the section, we give a family of digraphs with $\Omega(n^2)$ critical edges w.r.t. the two variants of URMT - synchronous URMT$_{LV}$ and asynchronous URMT$_{MC}$.

2 Model and Definitions

Network Model and Protocol: We model the underlying network 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 directed edges in the network. We assume the secure channels setting, i.e., all the edges are secure and authenticated. We also assume that every node is aware of the topology of the network. A sender $\mathbf{S} \in V$ and a receiver $\mathbf{R} \in V$ are two distinguished nodes in the network.

We assume that every node can be modelled as an interactive probabilistic Turing Machine. An interaction between a set of nodes in known as *protocol*. (This is a popular abstraction technique described at length in [3].) An *execution of a protocol* is defined as a run of a protocol with inputs and coin tosses from participating nodes.

Fault model: We model faults in the network by a fictitious centralized entity called the *adversary* which has unbounded computing power [9, 1]. A single "snapshot" of faults in the network can be described as a set of nodes $B \subseteq V \backslash \{\mathbf{S},\mathbf{R}\}$ [4], which means that all the nodes in B are faulty. We denote the set of all such B's by $\mathbb{A}$ and refer to it as an *adversary structure*. 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 maximal basis. An adversary structure $\mathbb{A}$ is t-threshold if every member of $\mathbb{A}$ is of size t; otherwise it is non-threshold. We only deal with cases where $|\mathbb{A}| \geq 2$, since otherwise the problems are trivial.

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. Additionally, we allow the adversary to be *adaptive* – it can choose which nodes to corrupt during an execution of a protocol based on its view, as long as the set of nodes corrupted during the entire execution is a member of $\mathbb{A}$.

[3] This turns out to be true for all the three variants of URMT discussed in this paper.
[4] We assume that $\mathbf{S}$ and $\mathbf{R}$ are non-faulty, for otherwise reliable message transmission need not happen.

We assume that the adversary knows the topology of the network as well as the protocol specification. We further make a conservative assumption that the adversary knows the message sender **S** has chosen to send to **R**. The results we prove in this paper hold good even if we do not make this assumption, but with a slight change in our definition of URMT (see [8]).

Timing model: Protocols running over directed networks tolerating an adversary rely heavily on the information of timing of various events within the system. We consider two extremes w.r.t timing model: all the edges in the network are either *synchronous* or *asynchronous*. Former case is referred to as synchronous networks and the latter as asynchronous networks.

In synchronous networks, 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 performs local computation on the received messages, in that order. Readers may find rigorous description of the model in [1].

In asynchronous networks, there is no fixed upper bound on the timing of events. In order to model computation in such networks, 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. Computation in such networks proceed in a sequence of steps, order of which is controlled by the adversary. In each step a single node is active. The node is activated by receiving a message; it then performs an internal computation, and possibly sends messages on its outgoing channels. For more details refer [2, 7].

2.1 Reliability

We refer to *Las Vegas* URMT as URMT_{LV} and *Monte Carlo* URMT as URMT_{MC}. We may also use URMT without any subscript to refer to both the variants together.

In the definitions that follow, probabilities are taken over the coin tosses of non-faulty nodes and the adversary. The message space is a large finite field $\langle \mathbb{F}, +, \cdot \rangle$ – all computations are done in this field.

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

Definition 2 ($(\mathbb{A}, \delta)$-URMT$_{LV}$). *Let $\delta < \frac{1}{2}$. We say that a protocol for transmitting messages in a network $\mathcal{N}$ from $\boldsymbol{S}$ to $\boldsymbol{R}$ is $(\mathbb{A}, \delta)$-URMT$_{LV}$ if for all valid Byzantine corruptions of any $B \in \mathbb{A}$ and $\forall \mathbf{m} \in \mathbb{F}$, the probability that $\boldsymbol{R}$ outputs $\boldsymbol{m}$ given that $\boldsymbol{S}$ has sent $\boldsymbol{m}$, is at least $(1 - \delta)$. Otherwise, $\boldsymbol{R}$ 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 $\boldsymbol{S}$ to $\boldsymbol{R}$ 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 $\boldsymbol{R}$ outputs $\boldsymbol{m}$ when $\boldsymbol{S}$ has sent $\boldsymbol{m}$ is* 1.

When $\mathbb{A}$ is a t-threshold adversary structure, we refer to $(\mathbb{A}, \delta)$-URMT and $\mathbb{A}$-PRMT as (t, δ)-URMT and t-PRMT respectively.

2.2 Preliminaries

In a directed network, besides the strong paths between sender and receiver, weak paths are also very useful in designing protocols [13] [5] .

Definition 4 (Strong path). *A sequence of nodes $v_1, v_2, v_3, \ldots, v_k$ is said to be a strong path from v_1 to v_k in the network $\mathcal{N} = (V, \mathcal{E})$ if for each $1 \leq i < k,\ (v_i, v_{i+1}) \in \mathcal{E}$.*

We assume that there vacuously exists a strong path from a node to itself.

Definition 5 (Weak path). *A sequence of nodes $v_1, v_2, v_3, \ldots, v_k$ 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 $(v_{i+1}, v_i) \in \mathcal{E}$.*

Along any weak path p, there are two special kinds of nodes:

- *Blocked node:* A node whose out-degree along p is 0.
- *Head node:* A node whose out-degree along p is 2 if it is an intermediate node, or 1 if it is a terminal node.

Consider a weak path p between **S** and **R**. If **S** is not a head node (i.e., it is a blocked node) along p, it can simulate two nodes **s** and **u**, and a directed edge (**s**,**u**). The incoming path to **S** along p (which made **S** a blocked node) becomes an incoming path to **u**, and the virtual sender **s** now acts as a head node. Analogously, it can always be ensured that **R** is a blocked node.

This allows us to view the path p as an alternating sequence of blocked nodes u_i's and head nodes y_i's starting with **S** as a head node denoted by y_0 and ending with **R** as a blocked denoted by u_{n+1}. In other words, the path p can be represented as $y_0, u_1, y_1, u_2, y_2, \ldots, u_n, y_n, u_{n+1}$ for some $n \geq 0$ such that y_0 has a strong path to u_1 along p, and y_i $(i > 0)$ has a strong path to u_i and u_{i+1} along p (see Figure 1). Such a representation of a weak path comes handy in giving easy to understand sufficiency proofs.

Message Authentication: Following [13], we define an *information-theoretically secure* message authentication scheme χ which is used extensively in our protocols. For any message $m \in \mathbb{F}$, $\chi(m; K_1, K_2, K_3) = (m + K_1, (m + K_1) \cdot K_2 + K_3)$, where $K_1, K_2, K_3 \in \mathbb{F}$ and are referred to as keys.

[5] Strong paths are usually just referred to as paths. Since we want to distinguish between two different kinds of paths, we use the adjectives 'strong' and 'weak' here.

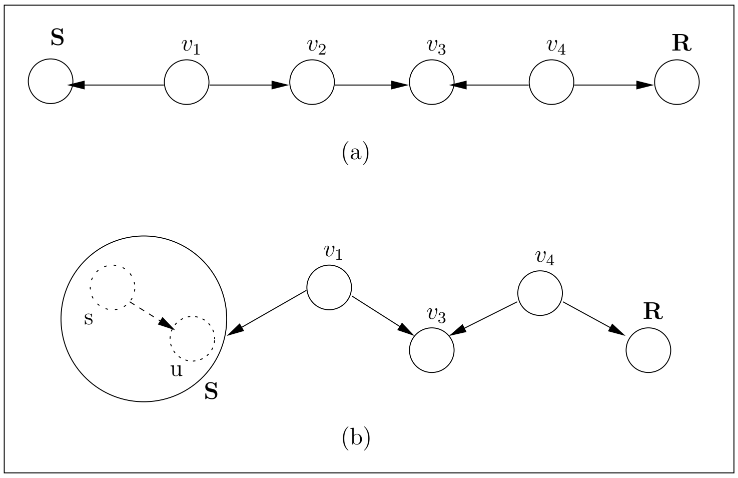

Fig. 1. (a) A weak path between **S** and **R**. (b) We view it as a sequence of alternating blocked nodes and head nodes starting with a head node **s** (the virtual sender) and ending in a blocked node **R**. Nodes **s**, v_1 and v_4 are head nodes; **u** and v_3 are blocked nodes such that there is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

If three randomly chosen keys (unknown to the adversary) are established between two nodes u and v such that there exists a strong path p (possibly containing some faulty nodes) 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 $\langle x', y' \rangle$; it verifies whether $y' \stackrel{?}{=} x' \cdot K_1 + K_2$. 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 [6]. Moreover, the view of nodes on the path p does not reveal any information about the message m.

3 Characterizing synchronous networks for $(\mathbb{A}, \delta)$-URMT$_{LV}$

In this section, we deal with the possibility and impossibility of Las Vegas URMT protocols from a sender **S** to a receiver **R** tolerating an adversary structure $\mathbb{A}$[7], when the underlying network can be abstracted as a directed graph, all its edges being synchronous. We refer to this variant of URMT as $(\mathbb{A}, \delta)$-URMT$_{LV}$, which was formally defined in Definition 2.

Since it is easier to deal with fixed size adversary structures, we first present the following reduction (similar reductions can be found in [14, 13]).

[6] Proofs for the same appear in [12].

[7] Recall that the adversary is non-threshold, adaptive and Byzantine.

Theorem 1. *In a directed synchronous network $\mathcal{N}$, an $(\mathbb{A}, \delta)$-URMT$_{LV}$ protocol exists if and only if for every adversary structure $\mathcal{A} \subseteq \mathbb{A}$ such that $|\mathcal{A}| = 2$, an $(\mathcal{A}, \delta)$-URMT$_{LV}$ protocol exists.*

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 a protocol tolerating an arbitrary sized adversary structure.

Let $f \in \mathbb{F}$ be any element **S** intends to send to **R**. Let $\mathcal{A}$ be any subset of $\mathbb{A}$ of size greater than 2. 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 an $(\mathcal{A}_i, \delta)$-URMT$_{LV}$ protocol. We use Y_i s as sub-protocols to construct a protocol Γ which is an $(\mathcal{A}, \delta)$-URMT$_{LV}$ protocol (as proved in the following lemma).

Firstly, by repeating Y_i sufficiently many times with the same message, we can amplify the probability of success to obtain an $(\mathcal{A}_i, \frac{\delta}{2})$-URMT$_{LV}$ protocol, say Z_i. The protocol Γ is now constructed 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; in case there is no majority, it outputs $\perp$.

□

Lemma 1. *For the directed synchronous network $\mathcal{N}$, the protocol Γ constructed above is an $(\mathcal{A}, \delta)$-URMT$_{LV}$ 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 outcomes of the two sub-protocols Z_2 and Z_3 are 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 δ. Additionally, it is easy to see that **R** would never output an incorrect message. □

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

Theorem 2. *In a directed synchronous network $\mathcal{N}$, $(\mathcal{A}, \delta)$-URMT$_{LV}$ protocol is possible if and only if for each $\alpha \in \{1, 2\}$, there exists a weak path q_α avoiding nodes in $B_1 \cup B_2$ such that every node u along the path q_α has a strong path to **R** avoiding all nodes in $B_{\overline{\alpha}}$[8]. (Paths q_1, q_2 need not be distinct.)*

We prove the theorem in the following sub-sections.

[8] $\overline{1} = 2$ and vice-versa.

3.1 Sufficiency

For a directed synchronous network $\mathcal{N}$, which satisfies the conditions given in Theorem 2, we show how to construct a protocol Π tolerating the adversary structure $\mathcal{A} = \{B_1, B_2\}$. Let m be the message **S** intends to send. If either q_1 or q_2 is a strong path from **S** to **R**, **S** trivially sends m along that path. When this is not the case, we construct two sub-protocols Π_1 and Π_2. For each $i \in \{1, 2\}$, sub-protocol Π_i uses the honest weak path q_i. We give a construction for Π_1 in Algorithm 1, and the construction of Π_2 follows by symmetry. For convenience of writing the protocol, we note that the weak path q_1 can be represented as $y_0, u_1, y_1, u_2, y_2, \ldots, u_n, y_n, u_{n+1}$ for some $n > 0$, where y_0 denotes **S** and u_{n+1} denotes **R**, as explained in Subsection 2.2.

1. **S** sends m to u_1 along q_1. For $1 \leq k \leq n$, node y_k chooses 3^k random keys namely $K_{k,1}, K_{k,2}, \ldots, K_{k,3^k}$ and sends those to u_k and u_{k+1} along q_1.

2. Node u_1 receives m from **S** and keys $K_{1,1}, K_{1,2}, K_{1,3}$ from y_1. It calculates $(\psi_{1,1}, \phi_{1,1}) = \chi(m; K_{1,1}, K_{1,2}, K_{1,3}) = (m + K_{1,1}, (m + K_{1,1}) \cdot K_{1,2} + K_{1,3})$ and sends it to **R** along a strong path avoiding B_2.
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 them to **R** along a strong path avoiding B_2. Formally, u_k calculates $(\psi_{k,j}, \phi_{k,j}) = \chi(K_{k-1,j}; K_{k,3j-2}, K_{k,3j-1}, K_{k,3j})$ for all $1 \leq j \leq 3^{k-1}$.

3. **R** receives $(\psi'_{k,j}, \phi'_{k,j})$, $1 \leq j \leq 3^{k-1}$, from the node u_k. If it does not receive a proper message from u_k, it concludes that 'B_1 is faulty' and stops. Additionally, **R** receives $\{K'_{n,1}, K'_{n,2}, \ldots, K'_{n,3^n}\}$ from y_n along q_1.

for k in n to 2 **do**
  R verifies whether $\phi'_{k,j} \stackrel{?}{=} \psi'_{k,j} \cdot K'_{k,3j-1} + K'_{k,3j}$ for all $1 \leq j \leq 3^{k-1}$. If the verification fails for any j, **R** concludes that 'B_1 is faulty' and stops. Otherwise, **R** recovers $K'_{k-1,j}$ as $\psi'_{k,j} - K_{k,3j-2}$ for every j.
end for

If at the end of the loop **R** has recovered $K'_{1,1}, K'_{1,2}, K'_{1,3}$, it verifies whether $\phi'_{1,1} \stackrel{?}{=} \psi'_{1,1} \cdot K'_{1,2} + K'_{1,3}$. If the verification passes, **R** recovers $m_1 = \psi'_{1,1} - K'_{1,1}$ as the message; otherwise, it concludes that 'B_1 is faulty'.

Algorithm 1: Sub-protocol Π_1

The sub-protocols Π_1 and Π_2 are run on the network $\mathcal{N}$. Based on the outcomes of these protocols, **R** takes one of the following actions:

- If **R** detects that B_i is corrupt in Π_i, it outputs the message $m_{\bar{i}}$ it recovered from $\Pi_{\bar{i}}$.
- If **R** recovers messages from each of the Π_i's and the messages are same, it outputs this message.
- If messages recovered through Π_1 and Π_2 are different, it outputs $\perp$.

This completes the description of Π.

Proof of Correctness: Since the weak path q_1 does not contain any faulty node, all the field elements send by the head nodes y_is in Step 1 are received reliably by the blocked nodes u_is. Moreover, the adversary does not gain any information about these elements. If the adversary corrupts B_1, it may affect the outcome of protocol Π_1 by changing one of the $(\psi_{k,j}, \phi_{k,j})$ sent by u_k to $\mathbf{R}$ along a path avoiding B_2, but not necessarily B_1. However, since the adversary has no knowledge of the keys $K_{k,3j-1}$ and $K_{k,3j}$, the probability that it can successfully generate a new tuple which passes the verification at $\mathbf{R}$ is atmost $\frac{1}{|\mathbb{F}|}$. In any case, a Byzantine corruption of B_1 does not affect the outcome of protocol Π_2.

Once the protocols Π_1 and Π_2 have terminated, we see how the decision rule at $\mathbf{R}$ produces the desired outcome with high probability:

- For some i, $\mathbf{R}$ concludes through Π_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 Π_i. Hence, if some verification fails during Π_i, B_i has to be faulty, and $\Pi_{\bar{i}}$ should recover the correct message m.
- For each $i \in \{1, 2\}$, all verifications in Π_i pass.
  - $m_i = m_{\bar{i}}$, $\mathbf{R}$ outputs m_i. Since one of m_i or $m_{\bar{i}}$ has to be same as m, $\mathbf{R}$'s output is correct.
  - $m_i \neq m_{\bar{i}}$. This implies that one of B_1 or B_2 was corrupt and managed to change one of the authenticated messages without being detected at $\mathbf{R}$. Since this happens with atmost $\frac{1}{|\mathbb{F}|}$ probability, $\mathbf{R}$ outputs $\perp$ with probability $\leq \frac{1}{|\mathbb{F}|}$.

Hence Π is an $(\mathcal{A}, \frac{1}{|\mathbb{F}|})$-URMT$_{LV}$ protocol.

3.2 Necessity

Let $\mathcal{N}$ be a network that does not satisfy the conditions of Theorem 2. We show that in such a network $(\{B_1, B_2\}, \delta)$-URMT$_{LV}$ 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[9]. Let the path q_1 be not present between $\mathbf{S}$ and $\mathbf{R}$ in $\mathcal{N}$ [10]. 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. For the sake of contradiction, let us assume that there exists a $(\{B_1, B_2\}, \delta)$-URMT$_{LV}$ protocol π in the network $\mathcal{N}$.

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$_{LV}$ from s^* to r^* is impossible in Lemma 2. We then show that the digraph $\mathcal{N}$ can be partitioned into disjoint sets whose connectivity properties are similar to the connectivity between

[9] In case $B_1 \cap B_2 \neq \phi$, adversary strategy to fail any protocol in $\mathcal{N}$ includes *fail-stopping* the nodes in the intersection.

[10] The case when the path q_2 is not present from $\mathbf{S}$ to $\mathbf{R}$ can be handled analogously.

nodes of digraph $\mathcal{N}^*$ in Lemma 3. Finally, in Lemma 4, we prove that if a $(\{B_1, B_2\}, \delta)$-URMT$_{LV}$ protocol π exists in the network $\mathcal{N}$, a $(\{\{b_1\}, \{b_2\}\}, \delta)$-URMT$_{LV}$ protocol π^* exists in the network $\mathcal{N}^*$, which is a contradiction. Hence, the conditions mentioned in Theorem 2 are necessary.

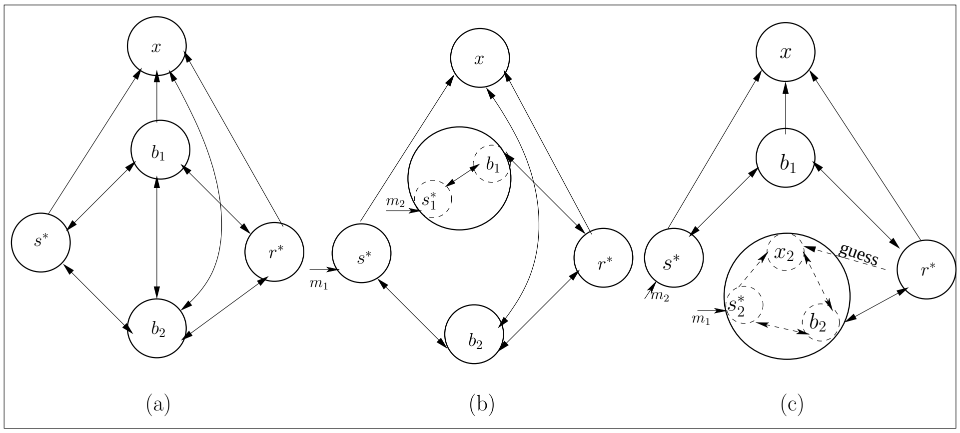

Fig. 2. (a) The directed network $\mathcal{N}^*$ (b) Adversary strategy when b_1 is faulty (c) Adversary strategy when b_2 is faulty

Lemma 2. *In the synchronous network $\mathcal{N}^*$ shown in Figure 2(a), $(\{\{b_1\}, \{b_2\}\}, \delta)$-URMT$_{LV}$ from s^* to r^* is impossible.*

Proof. Observe that the only weak path between s^* and r^* avoiding both b_1 and b_2 is the path formed by the sequence of nodes s^*, x, r^*. Node x is the only blocked node along this path and every path from it to $\mathbf{R}$ passes through b_2 – there is no path that avoids b_2. Hence this network does not satisfy the connectivity requirements of Theorem 2.

For the sake of contradiction, let us assume that a protocol ξ exists in $\mathcal{N}^*$ which is a $(\{\{b_1\}, \{b_2\}\}, \delta)$-URMT$_{LV}$ protocol. The protocol ξ is nothing but a collection of codes for the nodes in the network. Define ξ_{s^*} to be the code that s^* runs according to ξ, and in general define ξ_z to be the code that z runs according to ξ for any $z \in \{s^*, r^*, x, b_1, b_2\}$. These codes take different number of inputs and give different number of outputs depending on the in-degree and out-degree of the respective nodes. For instance, ξ_{s^*} expects two input messages, one from ξ_{b_1} and one from ξ_{b_2}, and gives three output messages, for ξ_{b_1}, ξ_{b_1} and ξ_x.

We first describe an adversary strategy $\mathcal{S}$ to fail any protocol ξ, and show how it works afterwards. Adversary chooses any two messages $m_1, m_2 \in \mathbb{F}$, $m_1 \neq m_2$.

When s^* intends to send m_i, adversary corrupts the node b_i [11] and snaps all communication with the nodes $b_{\bar{i}}$, x and s^*.

When adversary corrupts b_1, it simulates a system Υ_1 consisting of nodes b_1 and s_1^* as shown in the large circle in Figure 2(b). In Υ_1, the node b_1 runs the code ξ'_{b_1} and s_1^* runs the code ξ'_{s^*}, where ξ'_{b_1} and ξ'_{s^*} are same as ξ_{b_1} and ξ_{s^*} respectively, except that they always take NULL as input from b_2. Also, at the beginning of the execution, ξ'_{s^*} (the code running at s_1^*) is given the message m_2. The system Υ_1 is well defined and would continue to run as long as in every round ξ'_{b_1} can be provided with the inputs it expects from ξ_{r^*}. To achieve this, the message which r^* sends to b_2 in round $l-1$ is given as input to ξ'_{b_1} in the round l. Finally, whatever is output by ξ'_{b_1} intended for ξ_{r^*} in the round l is sent to r^*.

When adversary corrupts b_2, it simulates a system Υ_2 consisting of nodes b_2, x_2 and s_2^* as shown in the large circle in Figure 2(c). In Υ_2, the node b_2 runs the code ξ''_{b_2}, s_2^* runs the code ξ''_{s^*} and x_2 runs the code ξ''_x, where ξ''_{b_2}, ξ''_{s^*} and ξ''_x are same as ξ_{b_2}, ξ_{s^*} and ξ_x respectively, except that they always take NULL as input from b_1. Also, at the beginning of the execution, ξ''_{s^*} (the code running at s_2^*) is given the message m_1. The system Υ_2 is well defined and would continue to run as long as in every round ξ''_{b_2} and ξ''_x can be provided with the inputs they expect from ξ_{r^*}. To achieve this, the message which r^* sends to b_2 in round $l-1$ is given as input to ξ''_{b_2} in the round l. On the other hand, ξ''_x is given a *random message* as input from ξ_{r^*} in every round [12]. Finally, whatever is output by ξ''_{b_2} intended for ξ_{r^*} in the round l is sent to r^*.

With the adversary strategy $\mathcal{S}$, we show how the protocol ξ cannot be a valid $\{\{b_1\},\{b_2\}\}$-URMT$_{LV}$ protocol. In an execution E_i of the protocol ξ, let the coin tosses used by a code τ be denoted by $c_i(\tau)$. We consider two executions of ξ:

1. Execution E_1: s^* chooses to send m_1. Adversary corrupts b_1 and simulates the system Υ_1, as described earlier. r^* outputs m_1. [13]
2. Execution E_2: s^* chooses to send m_2. Adversary corrupts b_2 and simulates the system Υ_2. The coin tosses used by various codes in this execution are such that $c_2(\xi_{s^*}) = c_1(\xi'_{s^*})$, $c_2(\xi''_{s^*}) = c_1(\xi_{s^*})$, $c_2(\xi''_x) = c_1(\xi_x)$, $c_2(\xi_{b_1}) = c_1(\xi'_{b_1})$, $c_2(\xi''_{b_2}) = c_1(\xi_{b_2})$, and $c_2(\xi_{r^*}) = c_1(\xi_{r^*})$ [14]. Also, the *random message* input to ξ''_x in every round (as input from ξ_{r^*}) matches exactly with the message r^* sends to x in that round.

[11] Recall that we have assumed that the adversary knows the message sender chooses to send to the receiver.

[12] Note that, since node x is not corrupt, adversary does not have access to the messages r^* sends to x.

[13] For ξ to be a valid protocol, such an execution exists.

[14] Since x does not have a strong path to r^*, it does not have any effect on the outcome of the protocol. Hence its coin tosses do not matter.

The coin tosses of r^* as well as the messages received by it in execution E_2 are same as that in E_1. Hence r^* outputs m_1 in an execution where s^* chose to send message m_2, implying that ξ cannot be a valid URMT$_{LV}$ protocol. $\square$

Lemma 3. *The set of nodes V in the network $\mathcal{N}$ can be partitioned into 5 disjoint sets S^*, R^*, B_1', B_2 and X' such that $\mathbf{S} \in S^*$, $\mathbf{R} \in R^*$ and an edge exists from a node in $\boldsymbol{L}[i]$ to a node in $\boldsymbol{L}[j]$ only if $(\boldsymbol{l}[i], \boldsymbol{l}[j]) \in \mathcal{E}^*$ where $\boldsymbol{L} = [S^*, R^*, B_1', B_2, X']$ and $\boldsymbol{l} = [s^*, r^*, b_1, b_2, x]$ are two ordered lists, $\boldsymbol{l}[i]$ (resp. $\boldsymbol{L}[i]$) denotes the i^{th} element of the list $\boldsymbol{l}$ (resp. $\boldsymbol{L}$).*

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 R^*, S^* and X defined as follows: $R^* = \{w \mid w \in H$ and $\exists$ a weak path p between w and $\mathbf{R}$ s.t all the nodes in p have a strong path to $\mathbf{R}$ avoiding nodes in $B_2\}$; $S^* = \{w \mid w \in H \setminus R^*$ and w has a strong path to $\mathbf{R}$ avoiding $B_2\}$; and $X = H \setminus \{S^* \cup R^*\}$. 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. $B_1' = \{u \mid u \in B_1$ and u has a strong path to $\mathbf{R}$ avoiding $B_2\}$. $B_1^X = B_1 \setminus B_1'$. We consider the 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 that the only edges missing from $\mathcal{N}^*$ are $(x, s^*), (x, r^*), (x, b_1)$ and $(s^*, r^*), (r^*, s^*)$. $\square$

Lemma 4. *If a $(\{B_1, B_2\}, \delta)$-URMT$_{LV}$ protocol π exists from $\mathbf{S}$ to $\mathbf{R}$ in the network $\mathcal{N}$, a $(\{\{b_1\}, \{b_2\}\}, \delta)$-URMT$_{LV}$ protocol π^* exists from s^* to r^* in the network $\mathcal{N}^*$.*

This lemma can be proved using standard simulation techniques, hence we do not give a proof here.

From Lemma 2 we know that $(\{\{b_1\}, \{b_2\}\}, \delta)$-URMT$_{LV}$ is impossible from s^* to r^* in the network $\mathcal{N}^*$ – we arrive at a contradiction regarding the existence of π. Hence, the conditions mentioned in Theorem 2 are necessary.

4 Characterizing asynchronous networks for $(\mathbb{A}, \delta)$-URMT$_{MC}$

We now study the second variant of URMT – Monte Carlo URMT – in asynchronous networks. We refer to this variant as $(\mathbb{A}, \delta)$-URMT$_{MC}$, formally defined in Definition 1. In a manner similar to the previous section, we first provide a reduction that allows us to work with two-sized adversary structures.

Theorem 3. *In a directed asynchronous network $\mathcal{N}$, $(\mathbb{A}, \delta)$-URMT$_{MC}$ protocol is possible if and only if for every adversary structure $\mathcal{A} \subseteq \mathbb{A}$ such that $|\mathcal{A}| = 2$, $(\mathcal{A}, \delta)$-URMT$_{MC}$ protocol is possible.*

Proof. Necessity: Obvious. *Sufficiency:* The proof takes an approach similar to the one taken in the proof of Theorem 1. 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 1. For $i \in \{1, 2, 3\}$, let Z_i be an $(\mathcal{A}_i, \frac{\delta}{2})$-URMT$_{MC}$ protocol which can be constructed easily by repeating an $(\mathcal{A}_i, \delta)$-URMT$_{MC}$ protocol sufficiently many times, keeping **S**'s input same. The protocol η which is an $(\mathcal{A}, \delta)$-URMT$_{MC}$ protocol (as proved in the following lemma) is constructed as follows:

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

□

Lemma 5. *For the directed asynchronous network $\mathcal{N}$, the protocol η constructed above is an $(\mathcal{A}, \delta)$-URMT$_{MC}$ 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 Z_2 and Z_3 terminate with the correct output with at least $1 - \frac{\delta}{2}$ probability each. As **R** waits until two of the three Z_i sub-protocols terminate with same output, η fails only if at least one of Z_2 and Z_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, η is an $(\mathcal{A}, \delta - \frac{\delta^2}{4})$-URMT$_{MC}$, i.e., an $(\mathcal{A}, \delta)$-URMT$_{MC}$ protocol. □

Having reduced the problem of URMT$_{MC}$ in an asynchronous network tolerating an adversary structure to the problem of URMT$_{MC}$ tolerating all its 2-sized subsets, we now proceed to characterize directed asynchronous networks in which URMT$_{MC}$ tolerating adversary structure $\mathcal{A} = \{B_1, B_2\}$ is possible (where $B_1, B_2 \subseteq V \setminus \{\mathbf{S}, \mathbf{R}\}$).

Theorem 4. *In a directed asynchronous network $\mathcal{N}$, $(\mathcal{A}, \delta)$-URMT$_{MC}$ protocol is possible if and only if for each $\alpha \in \{1, 2\}$, there exists a weak path q_α avoiding nodes in $B_1 \cup B_2$ such that every node u along the path q_α has a strong path to **R** avoiding all nodes in $B_{\overline{\alpha}}$. (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 Π in Section 3.1. However, there are some important differences which will become evident in due course.

For a directed asynchronous network $\mathcal{N}$, which satisfies the conditions given in Theorem 4, we show how to construct a protocol ζ tolerating the adversary structure $\mathcal{A} = \{B_1, B_2\}$. Let m be the message **S** intends to send. If either q_1 or q_2 is a strong path from **S** to **R**, **S** trivially sends m along that path, and **R** is bound to receive it. When this is not the case, we construct two sub-protocols ζ_1 and ζ_2. For each $i \in \{1, 2\}$, sub-protocol ζ_i uses the honest weak path q_i. As usual, we give a construction of ζ_1 in Algorithm 2, and the construction of ζ_2 follows by symmetry. Once again, we note that the weak path q_1 can be represented as $y_0, u_1, y_1, \ldots, u_n, y_n, u_{n+1}$ for some $n > 0$, where y_0 denotes **S** and u_{n+1} denotes **R**, as explained in Subsection 2.2.

1. **S** sends m to u_1 along q_1. For $1 \le k \le n$, node y_k chooses 3^k random keys namely $K_{k,1}, K_{k,2}, \ldots, K_{k,3^k}$ and sends those to u_k and u_{k+1} along q_1.

2. Node u_1 waits for m to arrive from **S** and keys $K_{1,1}, K_{1,2}, K_{1,3}$ to arrive from y_1. It calculates $(\psi_{1,1}, \phi_{1,1}) = \chi(m; K_{1,1}, K_{1,2}, K_{1,3}) = (m + K_{1,1}, (m + K_{1,1}) \cdot K_{1,2} + K_{1,3})$ and sends it to **R** along a strong path avoiding B_2.

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

3. **R** waits for $\{K'_{n,1}, K'_{n,2}, \ldots, K'_{n,3^n}\}$ to arrive from y_n.

for k in n to 2 **do**

  R waits until it receives $\forall j \; 1 \le j \le 3^{k-1}$, $(\psi'_{k,j}, \phi'_{k,j})$ from u_k [2].

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

end for

If at the end of the loop **R** has recovered $K'_{1,1}, K'_{1,2}$ and $K'_{1,3}$, then **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, **R** recovers $m_1 = \psi'_{1,1} - K'_{1,1}$ as the message; otherwise it concludes that 'B_1 is faulty'.

Algorithm 2: Sub-protocol ζ_1

The sub-protocols ζ_1 and ζ_2 are run *in parallel* in the asynchronous network $\mathcal{N}$. Based on the outcomes of these protocols, **R** takes one of the following actions:

[1] As the weak path q_1 does not contain any faulty nodes, every u_k receives the field elements eventually.

[2] As these field elements are delivered along faulty paths, they may never arrive. However, since ζ_1 and ζ_2 are run in parallel (as mentioned in the sequel) and **R** waits for only one of them to terminate, the protocol ζ always terminates.

– For some $i \in \{1, 2\}$, if **R** detects that B_i is faulty during the run of ζ_i, it waits for $\zeta_{\bar{i}}$ to terminate and outputs $m_{\bar{i}}$ as the message.
– For some i, if **R** recovers m_i through ζ_i, it outputs that as the message without waiting for the sub-protocol $\zeta_{\bar{i}}$ to terminate.

This completes the description of ζ.

Proof of Correctness: We analyse the protocol case wise: (a) For some i, **R** detects that B_i is faulty during the run of ζ_i and outputs what it recovers from $\zeta_{\bar{i}}$. For each i, none of the nodes in B_i participate in the sub-protocol $\zeta_{\bar{i}}$. Therefore, some verification fails during ζ_i only if B_i is faulty. Hence, $\zeta_{\bar{i}}$ is bound to terminate with $m_{\bar{i}} = m$. (b) For some i, **R** recovers m_i through ζ_i. Probability that $m_i \neq m$ is at most $\frac{1}{|\mathbb{F}|}$. Hence, **R**'s output is correct with probability at least $\frac{|\mathbb{F}|-1}{|\mathbb{F}|}$. This implies that ζ is an $(\mathcal{A}, \frac{1}{|\mathbb{F}|})$-URMT$_{MC}$ protocol.

4.2 Necessity

Let $\mathcal{N}$ be an asynchronous network that does not satisfy the condition of Theorem 4. We show that in such a network $(\{B_1, B_2\}, \delta)$-URMT$_{MC}$ from **S** to **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 **S** and **R** in $\mathcal{N}$ (reasons for these assumptions were stated in Section 3.2). Hence, every weak path between **S** and **R** avoiding $B_1 \cup B_2$ has at least one node w such that every strong path from w to **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$_{MC}$ from s^* to r^* is impossible in $\mathcal{N}^*$ in Lemma 6. 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 3. Now, if $(\{B_1, B_2\}, \delta)$-URMT$_{MC}$ from **S** to **R** is possible in $\mathcal{N}$ then $(\{\{b_1\}, \{b_2\}\})$-URMT$_{MC}$ from s^* to r^* is possible in $\mathcal{N}^*$ (we need not prove this separately as the proof given in Lemma 4 works even when both $\mathcal{N}$ and $\mathcal{N}^*$ are asynchronous networks), which is a contradiction. This shows that no protocol for $(\{B_1, B_2\}, \delta)$-URMT$_{MC}$ can exist in the asynchronous network $\mathcal{N}$. Hence, the conditions mentioned in Theorem 4 are necessary.

Lemma 6. *In the asynchronous network $\mathcal{N}^*$ shown in Figure 2(a), $(\{\{b_1\}, \{b_2\}\}, \delta)$-URMT$_{MC}$ $(\delta < 1/2)$ from s^* to r^* is impossible.*

Proof. For the sake of contradiction, let us assume that a protocol τ exists in $\mathcal{N}^*$ which is a $(\{\{b_1\}, \{b_2\}\}, \delta)$-URMT$_{MC}$ protocol from s^* to r^*. Following the proof of Lemma 2, we define τ_z to be the code that node z runs according to τ, for $z \in \{s^*, r^*, x, b_1, b_2\}$. We describe an adversary strategy to fail protocol τ. Firstly, adversary *fixes* a schedule $\mathcal{D}$ – messages in the network are always exchanged according to this schedule (since the network is asynchronous, adversary can do this). It chooses any two messages $m_1, m_2 \in \mathbb{F}, m_1 \neq m_2$. When s^* intends to send m_i, it corrupts the node b_i [15], for $i \in \{1, 2\}$.

When adversary corrupts b_2, it simply fail-stops this node. Since τ is a URMT$_{MC}$ protocol, when s^* chooses to send m_2 and the node b_2 is fail-stopped, there must exist a finite time instant T such that r^* outputs m_2 before instant T with probability at least 1/2. Now, when adversary corrupts b_1, it delays all the outgoing messages from b_2 beyond the time instant T. Additionally, it simulates a system Υ consisting of nodes b_1 and s_1^* (in a manner similar to the system Υ_1 in Lemma 2). In Υ, the node b_1 runs the code τ'_{b_1} and s_1^* runs the code τ'_{s^*}, where τ'_{b_1} and τ'_{s^*} are same as τ_{b_1} and τ_{s^*} respectively, except that they always take NULL as input from b_2. Also, at the beginning of the execution, τ'_{s^*} (the code running at s_1^*) is given the message m_2. The system Υ is well defined and would continue to run as long as τ'_{b_1} can be provided with the inputs it expects from τ_{r^*}. To achieve this, the message which r^* sends to b_1 is given as input to τ'_{b_1}. Finally, whatever is output by τ'_{b_1} intended for τ_{r^*} is sent to r^*.

Let us see how the adversary strategy described above succeeds. When s^* chooses to send message m_2, adversary fail-stops b_2; this also *cuts-off* node x from the network. Therefore, only the messages generated by s^* and b_1 influence the output of r^*. Nonetheless, r^* outputs m_2 before time instant T with probability at least 1/2. On the other hand, when s^* chooses to send message m_1, adversary delays all the outgoing messages from b_2 beyond the time instant T, and simulates the system Υ as described above. As a result, the nodes s^*, x and b_2 are *cut-off* from the network till time T. Hence, only the messages generated by s_1^* and b_1, which are part of the system Υ, constitute the view of r^* till time T. Now, since the code τ'_{s^*} running at s_1^* was given the message m_2, the view of r^* in this case is indistinguishable from the previous case till time T. Therefore, r^* outputs m_2 before time T with probability at least 1/2 in this case as well.

Hence, τ cannot be a valid URMT$_{MC}$ protocol. □

We now state the main result of this paper: synchronous Las Vegas protocols are possible if and only if asynchronous Monte Carlo protocols are.

Corollary 1. *In a directed network $\mathcal{N} = (V, \mathcal{E})$, a synchronous $(\mathbb{A}, \delta)$-URMT$_{LV}$ protocol exists if and only if a protocol exists for asynchronous $(\mathbb{A}, \delta)$-URMT$_{MC}$.*

Proof. Follows from Theorem 1, 2 and 3, 4. □

5 Characterizing asynchronous networks for $(\mathbb{A}, \delta)$-URMT$_{LV}$

In this section we come back to the Las Vegas variant of URMT, this time in asynchronous networks though. As has been the case so far, we can show that working with a two-sized adversary structure is sufficient.

[15] Recall that we have assumed that the adversary knows the message sender chooses to send to the receiver.

Theorem 5. *In a directed asynchronous network $\mathcal{N}$, $(\mathbb{A}, \delta)$-URMT$_{LV}$ protocol is possible if and only if for every adversary structure $\mathcal{A} \subseteq \mathbb{A}$ such that $|\mathcal{A}| = 2$, $(\mathcal{A}, \delta)$-URMT$_{LV}$ protocol is possible.*

Proof. Similar to the proof of Theorem 3, hence omitted. □

For $\mathcal{A} = \{B_1, B_2\}$, where $B_1, B_2 \subseteq V \setminus \{\mathbf{S}, \mathbf{R}\}$, we have the following characterization.

Theorem 6. *In a directed asynchronous network $\mathcal{N}$, $(\mathcal{A}, \delta)$-URMT$_{LV}$ protocol is possible if and only if there exists a strong path from $\mathbf{S}$ to $\mathbf{R}$ avoiding nodes in $B_1 \cup B_2$.*

Proof. Sufficiency: Let m be the message $\mathbf{S}$ intends to send. Send m to $\mathbf{R}$ along the strong path avoiding nodes in $B_1 \cup B_2$. Since, the path does not contain any corrupt nodes, m is eventually received by $\mathbf{R}$. □

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 6. We first consider the 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$_{LV}$ from s^* to r^* is impossible in Lemma 7. 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 8. Finally in Lemma 9, we show that if $(\{B_1, B_2\}), \delta)$-URMT$_{LV}$ from $\mathbf{S}$ to $\mathbf{R}$ is possible in $\mathcal{N}$ then $(\{\{b_1\}, \{b_2\}\})$-URMT$_{LV}$ from s^* to r^* is possible in $\mathcal{N}_1^*$, which is a contradiction. Hence, the conditions mentioned in Theorem 6 are necessary.

Lemma 7. *In the asynchronous network $\mathcal{N}_1^*$, $(\{\{b_1\}, \{b_2\}\}, \delta)$-URMT$_{LV}$ from s^* to r^* is impossible.*

Proof. We assume that a protocol κ exists in $\mathcal{N}_1^*$ which is a $(\{\{b_1\}, \{b_2\}\}, \delta)$-URMT$_{LV}$ protocol from s^* to r^*. Once again, following the proof of Lemma 2, we define κ_z to be the code that node z runs according to κ, for $z \in \{s^*, r^*, b_1, b_2\}$. Also, in an execution E_i of the protocol κ, let the coin tosses used by a code τ be denoted by $c_i(\tau)$.

We describe an adversary strategy to fail any protocol κ. Firstly, adversary *fixes* a schedule $\mathcal{D}$ – messages in the network are always exchanged according to this schedule (since the network is asynchronous, adversary can do this). It chooses any two messages $m_1, m_2 \in \mathbb{F}$, $m_1 \neq m_2$. When s^* intends to send m_i, adversary corrupts the node b_i, for $i \in \{1, 2\}$. When adversary corrupts b_2, it simply fail-stops this node. Since κ is a URMT$_{LV}$ protocol, there must exist an execution where s^* chooses to send m_2, the node b_2 is fail-stopped, still r^*

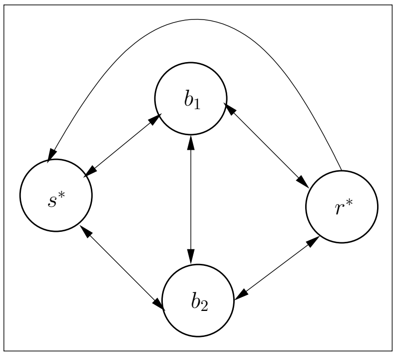

Fig. 3. The directed network $\mathcal{N}_1^*$

outputs m_2. Let this execution be E_2 (according to our naming convention, the coin tosses of the codes running at s^* and b_1 in this execution are $c_2(\kappa_{s^*})$ and $c_2(\kappa_{b_1})$), and let the time instant at which r^* output m_2 be T.

Now, when adversary corrupts b_1, it delays all the outgoing messages from b_2 beyond the time instant T. Additionally, it simulates a system Δ consisting of nodes b_1 and s_1^*. In Δ, the node b_1 runs the code κ'_{b_1} and s_1^* runs the code κ'_{s^*}, where κ'_{b_1} and κ'_{s^*} are same as κ_{b_1} and κ_{s^*} respectively, except that they always take NULL as input from b_2. The code κ'_{s^*} is given random messages as input from κ_{r^*} since the messages sent by r^* to s^* are not available to the adversary. Also, whatever is output by κ'_{b_1} intended for κ_{r^*} is sent to r^*. Importantly, at the beginning of the execution, κ'_{s^*} (the code running at s_1^*) is given the message m_2 as input.

Let us consider an execution E_1 where s^* chooses to send m_1 – hence adversary corrupts b_1, the coin tosses of codes running at simulated nodes s_1^* and b_1 (in the system Δ) are $c_2(\kappa_{s^*})$ and $c_2(\kappa_{b_1})$ respectively, and the random messages given to κ'_{s^*} (the code running at s_1^*) as input from κ_{r^*} match exactly with the messages r^* sends to s^*. Since the adversary delays all the outgoing messages from b_2 beyond the time instant T, r^* fails to distinguish whether it is running in E_1 or E_2. Since it outputs m_2 at time T in execution E_2, it also outputs m_2 at time T in execution E_1, where s^* chose to send m_1.

Hence, κ is not a valid URMT$_{LV}$ protocol. $\square$

We now consider network $\mathcal{N} = (V, \mathcal{E})$ which does not satisfy the conditions of Theorem 6.

Lemma 8. *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 $\boldsymbol{L}[i]$ to a node in $\boldsymbol{L}[j]$ only if $(\boldsymbol{l}[i], \boldsymbol{l}[j]) \in \mathcal{E}_1^*$, where $\boldsymbol{L} = [S^*, B_1, B_2, R^*]$ and $\boldsymbol{l} = [s^*, b_1, b_2, r^*]$ are two ordered lists, $\boldsymbol{l}[i]$ (resp. $\boldsymbol{L}[i]$) denotes the i^{th} element of the list $\boldsymbol{l}$ (resp. $\boldsymbol{L}$).*

Proof. We partition the non-faulty nodes $H = V \setminus \{B_1 \cup B_2\}$ 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 $S^* = V \setminus \{R^* \cup B_1 \cup B_2\}$. It is clear that $\mathbf{R} \in R^*$ and $\mathbf{S} \in S^*$. Moreover, any node $u \in S^*$ cannot have an edge to a node in R^*, otherwise u will become a part of R^*. Since the only edge missing from $\mathcal{N}_1^*$ is (s^*, r^*), we are done. □

Lemma 9. *In the directed asynchronous network $\mathcal{N} = (V, \mathcal{E})$, $(\{B_1, B_2\}, \delta)$-URMT$_{LV}$ is possible from $\mathbf{S}$ to $\mathbf{R}$ only if $(\{\{b_1\}, \{b_2\}\}, \delta)$-URMT$_{LV}$ is possible from s^* to r^* in the network $\mathcal{N}_1^*$.*

Proof. Proof is on the lines similar to the proof of Lemma 4, hence omitted.

From Lemma 7 we know that $(\{\{b_1\}, \{b_2\}\}, \delta)$-URMT$_{LV}$ is impossible from s^* to r^* in the network $\mathcal{N}_1^*$. Using Lemma 9, we arrive at a contradiction. Hence, the conditions mentioned in Theorem 6 are necessary.

We now present the second main result of this paper: the minimum connectivity requirements for the case of asynchronous Las Vegas protocols is same as that for perfect protocols.

Theorem 7. *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 [6]. □

Corollary 2. *In a directed network $\mathcal{N} = (V, \mathcal{E})$, an asynchronous $(\mathbb{A}, \delta)$-URMT$_{LV}$ protocol exists if and only if a protocol exists for synchronous (or asynchronous) $\mathbb{A}$-PRMT.*

Proof. Follows from Theorem 5, 6 and 7. □

6 Critical edges

We say that an edge is *critical* if its removal renders the graph insufficiently connected for a certain kind of protocol, though before its removal the connectivity was sufficient. While it is known that for perfect protocols the number of critical edges is always $O(n)$, we give a family of digraphs with $\Omega(n^2)$ critical edges for Las Vegas and Monte Carlo variants, which have less connectivity requirements!

In [13], Bhavani et al. proposed a family of digraphs for synchronous (t, δ)-URMT$_{MC}$ with the same lower bound on the number of critical edges. However, their claim is incorrect as shown in the following theorem. For convenience, we first state their family of graphs G_t $(t > 0)$ here: $G_t = (V, \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3)$ where $V = \{\mathbf{S}, v_1, \ldots, v_{t+1}, u_1, \ldots, u_t, \mathbf{R}\}$; $\mathcal{E}_1 = \bigcup_1^{t+1}\{(\mathbf{S}, v_i), (v_i, \mathbf{R})\}$; $\mathcal{E}_2 = \bigcup_{i=1}^{t}\{(\mathbf{S}, u_i), (\mathbf{R}, u_i)\}$; and $\mathcal{E}_3 = \bigcup_{i=1}^{t}\{(u_i, v_1), \ldots, (u_i, v_{t+1})\}$ (see Figure 4(a)).

Theorem 8. *G_t has only $\Theta(n)$ critical edges w.r.t synchronous (t, δ)-URMT$_{MC}$.*

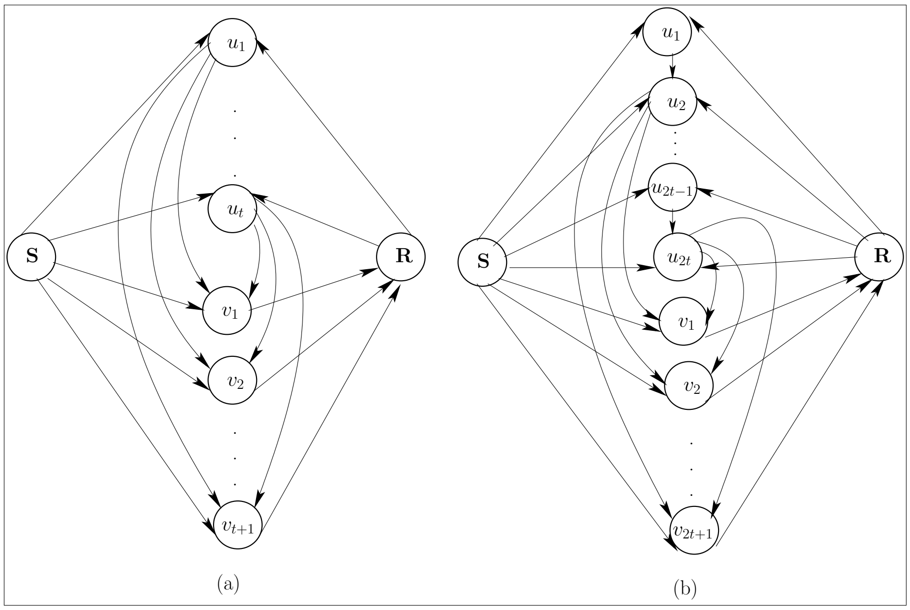

Fig. 4. (a) Graph G_t, (b) Graph H_t

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 [13], for (t,δ)-URMT$_{MC}$ to be impossible in G'_t, there exist B_1, B_2 such that $|B_1|, |B_2| \leq t$, and every weak path from **S** to **R** avoiding nodes in $B_1 \cup B_2$ has at least one node x such that every strong path from x to **R** passes through nodes some nodes in B_1 as well as some nodes in B_2. We first show that no such sets B_1, B_2 exist for G'_t.

If there exists a strong path from **S** to **R** avoiding nodes in $B_1 \cup B_2$, URMT$_{MC}$ 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. Hence the path $\mathbf{S} \rightarrow u_i \leftarrow \mathbf{R}$ is an honest weak path from **S** to **R**, with u_i having a strong path to **R** via v_i. For the impossibility of synchronous (t,δ)-URMT$_{MC}$, 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 $B_1 = B_2 = \{v_{\alpha_1}, v_{\alpha_2}, \ldots, v_{\alpha_t}\}$ for some $\{\alpha_1, \alpha_2, \ldots, \alpha_t\} \subset \{1, 2, 3, \ldots, t+1\}$. But this leaves a strong honest path from **S** to **R**. Hence, no B_1, B_2 exist such that (t,δ)-URMT$_{MC}$ is impossible in G'_t, i.e., (t,δ)-URMT$_{MC}$ 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[13] 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 **S** and **R**, G_t has $\Omega(n)$ critical edges. It therefore follows that G_t has $\Theta(n)$ critical edges. $\square$

We now propose a family of graphs with $\Omega(n^2)$ critical edges. For all $t > 0$, consider $H_t = (\mathrm{V}^1, \bigcup_{i=1}^{4} \mathcal{E}_i^1)$ with $\mathrm{V}^1 = \{\mathbf{S}, v_1, \ldots, v_{2t+1}, u_1, \ldots, u_{2t}, \mathbf{R}\}$; $\mathcal{E}_1^1 = \bigcup_1^{2t+1} \{(\mathbf{S}, v_i), (v_i, \mathbf{R})\}$; $\mathcal{E}_2^1 = \bigcup_{i=1}^{2t} \{(\mathbf{S}, u_i), (\mathbf{R}, u_i)\}$; $\mathcal{E}_3^1 = \bigcup_{i=1}^{t} \{(u_{2i-1}, u_{2i}))\}$; and $\mathcal{E}_4^1 = \bigcup_{i=1}^{t} \{(u_{2i}, v_1), \ldots, (u_{2i}, v_{2t+1})\}$, as shown in Figure 4(b). Here, number of nodes in graph H_t is $n = 4t + 3$.

Theorem 9. *H_t has $\Omega(n^2)$ critical edges w.r.t synchronous $(2t, \delta)$-URMT$_{MC}$.*

Proof. $(2t, \delta)$-URMT$_{MC}$ is possible in H_t (follows from [13]). 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$_{MC}$ impossible, hence $(2t, \delta)$-URMT$_{MC}$ is impossible. Therefore, H_t has $\Omega(|\mathcal{E}_3^1|)$ or $\Omega(n^2)$ critical edges.

6.1 Critical Edges for asynchronous Monte Carlo and synchronous Las Vegas

In this section we show the existence of a family of digraphs with $\Omega(n^2)$ critical edges w.r.t asynchronous (t, δ)-URMT$_{MC}$ and synchronous (t, δ)-URMT$_{LV}$ – the family of digraphs G_t presented in [13] turns out to be the family we desire! Since the characterization of synchronous networks for the possibility of URMT$_{LV}$ is same as that of asynchronous networks for URMT$_{MC}$ (proved in Corollary 1), we can deduce that any given graph (meeting the sufficiency conditions) has same number of critical edges w.r.t both the aforementioned variants. Hence, a family of digraph with $\Omega(n^2)$ critical edges w.r.t asynchronous (t, δ)-URMT$_{MC}$ has the same bound on critical edges w.r.t synchronous (t, δ)-URMT$_{LV}$ too.

Theorem 10. *G_t has $\Omega(n^2)$ critical edges w.r.t asynchronous (t, δ)-URMT$_{MC}$ (and synchronous (t, δ)-URMT$_{LV}$).*

Proof. Asynchronous (t, δ)-URMT$_{MC}$ is possible in G_t (follows from Theorem 3, 4). 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$_{MC}$ impossible, hence (t, δ)-URMT$_{MC}$ is impossible – therefore, $\Omega(|\mathcal{E}_3|)$ or $\Omega(n^2)$ critical edges. □

References

1. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation. In *Proceedings of the 20th Symposium on Theory of Computing (STOC)*, pages 1–10. ACM Press, 1988.
2. 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.

3. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of the 42nd Symposium on Foundations of Computer Science (FOCS)*, pages 136–145. IEEE, 2001. Full version available at `http://eprint.iacr.org/2000/067`.
4. Ashish Choudhary, Arpita Patra, B. V. Ashwinkumar, Kannan Srinathan, and C. Pandu Rangan. On minimal connectivity requirement for secure message transmission in asynchronous networks. In *ICDCN '09: Proceedings of the 10th International Conference on Distributed Computing and Networking*, pages 148–162, Berlin, Heidelberg, 2009. Springer-Verlag.
5. 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.
6. D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly Secure Message Transmission. *Journal of the Association for Computing Machinery (JACM)*, 40(1):17–47, January 1993.
7. Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
8. 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.
9. M. Hirt and U. Maurer. Player Simulation and General Adversary Structures in Perfect Multi-party Computation. *Journal of Cryptology*, 13(1):31–60, April 2000.
10. Arpita Patra, Ashish Choudhary, and C. Rangan. Unconditionally reliable and secure message transmission in directed networks revisited. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *Security and Cryptography for Networks*, volume 5229 of *Lecture Notes in Computer Science*, pages 309–326. Springer Berlin / Heidelberg, 2008.
11. Arpita Patra, Ashish Choudhary, and Chanrasekharan Pandu Rangan. Constant phase efficient protocols for secure message transmission in directed networks. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, PODC '07, pages 322–323, New York, NY, USA, 2007. ACM.
12. 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.
13. 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.
14. 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.
15. Qiushi Yang and Yvo Desmedt. Cryptanalysis of secure message transmission protocols with feedback. In Kaoru Kurosawa, editor, *Information Theoretic Security*, volume 5973 of *Lecture Notes in Computer Science*, pages 159–176. Springer Berlin / Heidelberg, 2010.