Unconditionally Reliable Message Transmission in Directed Neighbour Networks

Shashank Agrawal[*] Abhinav Mehta[*] Kannan Srinathan[*]
{shashank.agrawal@research. abhinav_mehta@research. srinathan@}iiit.ac.in

Abstract

The problem of *unconditionally reliable message transmission* (URMT) is to design a protocol which when run by players in a network enables a sender **s** to deliver a message to a receiver **r** with high probability, even when some players in the network are under the control of an *unbounded* adversary. Renault and Tomala [JoC2008] gave a characterization of undirected neighbour networks over which URMT tolerating Byzantine adversary is possible. In this paper, we generalize their result to the case of directed networks.

1 Introduction

In Distributed Computing literature, it is widely assumed that every pair of participating players/nodes in a communication network are connected by a reliable channel. However, in practice, most of the players in a network are not directly connected to each other. In particular, if a sender **s** wishes to send a message reliably to a receiver **r**, it may have to route messages through intermediate nodes some of whom may be faulty. The problem of *unconditionally reliable message transmission*(URMT) is to design a protocol which when run by the players in a network helps in the simulation of a highly reliable channel from **s** to **r**, even in the presence of these faults.

URMT has been studied in various network, timing, and fault models. In particular, unicast networks have been very well studied in literature. In [2], Desmedt and Wang give efficient protocols for URMT (which additionally provide perfect secrecy) tolerating Byzantine adversary in directed networks by abstracting the network as a collection of vertex-disjoint paths between **s** and **r**. In [10], Srinathan et. al. forgo this abstraction and characterize general directed networks in which URMT tolerating *mixed* adversary is possible. While the two papers discussed above assume network to be synchronous, the more general case of asynchronous network is studied in [1].

We consider a special kind of multi-recipient network called neighbour network, first introduced in [4]. In a neighbour network, when a player sends a message, it is received by all of its neighbours, i.e., every player shares a multicast channel with its neighbours in some underlying graph. Multi-recipient networks arise frequently in practice; for instance, a local area network like an Ethernet bus or a token ring. Other examples include the Bluetooth or IEEE 802.11 network. Yet another motivation for the study of multicast models comes from game theory as described in [8].

In [3], Franklin and Wright initiate the study of URMT in undirected neighbour networks abstracting it as a collection of neighbour-disjoint multicast lines between **s** and **r**. They show that $t + 1$ multicast lines are sufficient for URMT tolerating t-threshold Byzantine adversary. When compared to a unicast network where $2t + 1$ vertex-disjoint paths are necessary, this is a significant improvement. In [8], Renault and Tomala extend this characterization to general

[*]Center for Security, Theory and Algorithmic Research (C-STAR), International Institute of Information Technology, Hyderabad, 500032, India.

undirected neighbour networks. The case of directed neighbour networks was left as an open problem.

In this work, we give a characterization of directed neighbour networks over which URMT tolerating non-threshold Byzantine adversary (defined formally later) is possible. Our model is essentially the same as [8] with the important difference that their underlying network is undirected while ours is directed. Note that, for several real-life networks where a node can communicate with another node but not the other way round, undirected graphs are not a suitable model. For instance, in a sensor network where different nodes have different transmission power, communication links tend to be uni-directional: a node u can hear v but v cannot hear u [11]. Moreover, as directed networks are a strict generalization, our results can be adapted to the undirected case also.

2 Organization

In section 3 we describe our model and define URMT formally. In section 4, we show by means of an example why the approach taken in the case of undirected neighbour networks may not be applicable to their directed counterpart. In section 5, we first show how the existence of a URMT protocol tolerating an arbitrary adversary structure $\mathbb{A}$ reduces to the existence of URMT protocols tolerating every two-sized subset of $\mathbb{A}$. We then focus on giving a characterization of networks over which URMT tolerating a fixed two-sized adversary structure is possible. In Theorem 4, we give a necessary and sufficient condition on directed neighbour networks for a URMT protocol to exist. To prove sufficiency, we give a protocol for URMT in any network that satisfies conditions of Theorem 4. In the following section 6, we show that the condition is indeed necessary. Our last section 7 concludes the paper by summarising the results and discussing a few related open problems.

3 Model and Definitions

We model the neighbour network as a digraph $G = (V, E)$, where V denotes the set of nodes (or players) in the network and $E \subseteq V \times V$ represents the set of secure and authenticated channels available between nodes (similar to the *secure channels* setting). Two special nodes $\mathbf{s}, \mathbf{r} \in V$ denote the sender and receiver respectively. The system is assumed to be synchronous, that is, the protocol is executed in a sequence of *rounds* [8]. Whenever a node sends a message, it is identically received by all its out-neighbours. We assume that the topology of the network (i.e. G) is known to every node.

Fault in the network is modelled via an unbounded centralized fictitious entity called the adversary that can control a subset of nodes in the network and make them behave in a Byzantine fashion [6]. 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 $\mathbf{r}$ [1].

The subset of nodes which an adversary can control is specified via an adversary structure. Formally, an (non-threshold) adversary structure $\mathbb{A}$ is a *monotone* set of subsets of the node set, i.e., $\mathbb{A} \subseteq \mathcal{P}(V \setminus \{\mathbf{s}, \mathbf{r}\})$ and if $A \in \mathbb{A}$ then all subsets of A also belong to $\mathbb{A}$ [5]. During an execution, adversary may corrupt the nodes of one particular set in $\mathbb{A}$. For a structure $\mathbb{A}$, $\overline{\mathbb{A}}$ denotes the maximal basis of $\mathbb{A}$ given by $\overline{\mathbb{A}} = \{A \in \mathbb{A} \mid \nexists\, A' \in \mathbb{A} \text{ s.t. } A \subset A'\}$. We assume that

[1] 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 [3]).

$\mathbb{A}$ itself is a maximal basis. In the sequel when we talk about an adversary structure, it will be clear from the context which graph is it referring to.

Throughout this paper, nodes are denoted in lowercase while set of nodes in uppercase. A node which can never be corrupt is called honest, i.e., a node $u \in V$ is honest if $\nexists$ an $X \in \mathbb{A}$ s.t. $u \in X$. A path C in a graph G is a finite sequence of nodes $C = (c_1, c_2, \ldots, c_n)$ s.t. $\forall\, 1 \leq i < n\ (c_i, c_{i+1}) \in E$. A path is an honest path if every c_i is honest. For $u, v \in V$, we say that C is a path from u to v if $c_1 = u$ and $c_n = v$. We refer to the set of out-neighbours of a node u including u itself as the neighbourhood of u, $N(u) = \{v : \exists\, (u, v) \in E\} \cup \{u\}$. Neighbourhood of a set of nodes X is defined as $N(X) = \bigcup_{u \in X} N(u)$.

Let the message space be a large finite field $\langle \mathbb{F}, +, \cdot \rangle$. All computations are done in this field. We now formally define URMT as well as a weaker form of it, URMT$_F$. Probabilities are taken over the random inputs of all honest players and the random inputs of the adversary.

Definition 1 (($\mathbb{A}, \delta$)-URMT)**.** *Let $\delta < \frac{1}{2}$. We say that a protocol for transmitting a message in a synchronous network G from **s** to **r** is $(\mathbb{A}, \delta)$-URMT if for all valid Byzantine corruptions of any $A \in \mathbb{A}$ and for every $m \in \mathbb{F}$, the probability that **r** outputs m, given that **s** has sent m, is at least $(1 - \delta)$.*

Definition 2 (($\mathbb{A}, \delta$)-URMT$_F$)**.** *Let $\delta < \frac{1}{2}$. We say that a protocol for transmitting a message in a synchronous network G from **s** to **r** is $(\mathbb{A}, \delta)$-URMT$_F$ if for all valid Byzantine corruptions of any $A \in \mathbb{A}$ and for every $m \in \mathbb{F}$, the probability that **r** outputs m or knows that the set A is faulty, given that **s** has sent m, is at least $(1 - \delta)$.*

Suppose a node u wishes to send a message $f \in \mathbb{F}$ along a path C to another node v. If the adversary corrupts some nodes on path C, it can modify messages sent along the path. We want that the node v should be able to detect any modification with high probability. We achieve this by means of an authentication function χ which authenticates any message $m \in \mathbb{F}$ with a pair of keys K_1, K_2 to produce a 2-tuple $(m, m \cdot K_1 + K_2)$. Assuming a pair of random keys K', K'' unknown to the adversary has been established between u and v, instead of simply sending f along path C, the node u now sends $\chi(f; K', K'')$. Let v receive the 2-tuple (x, y); it verifies whether $y \overset{?}{=} x \cdot K' + K''$. If verification succeeds v outputs x, if it doesn't v knows that path C is faulty. It can be shown that if $x \neq f$ (i.e., the message has been modified) verification fails with probability atleast $1 - \frac{1}{|\mathbb{F}|}$ [7].

4 A Motivating Example

Consider the digraph $G_e = (V_e, E_e)$ shown in Figure 1. Let $G'_e = (V_e, E'_e)$ be the undirected counterpart of digraph G_e, where for $u, v \in V_e$, $(u, v) \in E'_e$ only if $(u, v) \in E_e$ or $(v, u) \in E_e$. For both the graphs, let $B_1 = \{b_1, b'_1, b''_1\}$, $B_2 = \{b_2, b'_2, b''_2\}$, and let the adversary structure be $\mathcal{A} = \{B_1, B_2\}$. By Definition 3.7 in [8], in the undirected graph G'_e, URMT is possible between **s** and p as well as between p and **r** (also between **s** and x as well as between x and **r**). If the sender **s** wishes to send a message m to the receiver **r**, as described in section 3.1 in [8], a URMT protocol (called the ϵ-Reliable Protocol) is first run between **s** and p on the message m and then between p and **r**, again on the message m.

However, in the digraph G_e, it is easy to see that URMT from p to **r** is impossible (node b'_1 may fail-stop), neither is URMT possible from x to **r**. Yet a protocol for URMT from **s** to **r** exists (proved formally later). Hence the approach taken in [8] is unlikely to work here. Nodes p and x can only do a weaker form of URMT, URMT$_F$, to **r**. As described in the definition

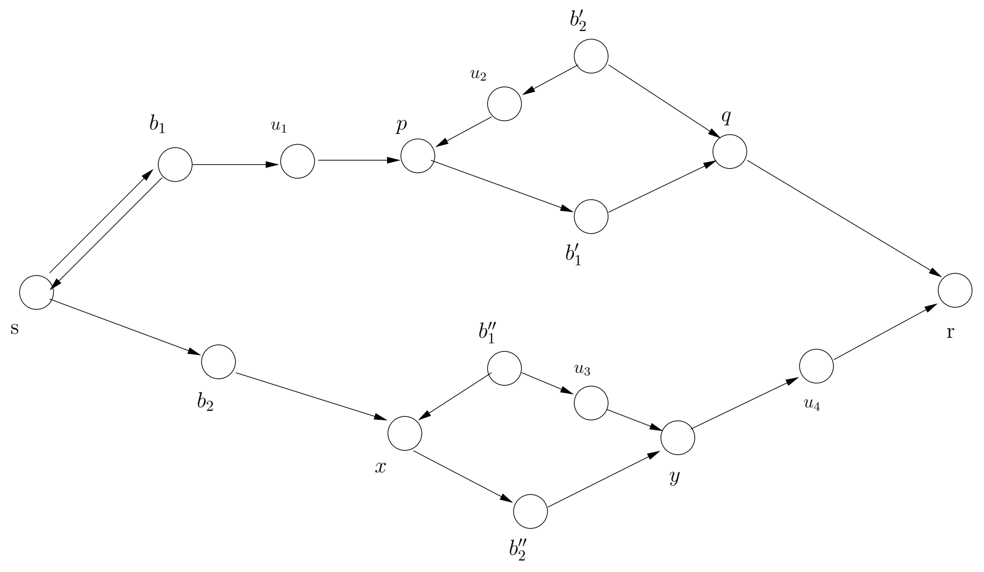

Figure 1: An Example Graph G_e

section, in the case of URMT$_F$, **r** must output the correct message or know the identity of the corrupt set with high probability.

5 Unconditionally Reliable Message Transmission

It is difficult and non-intuitive to work with an adversary structure of arbitrary size. In the following theorem, we show how the existence of a URMT protocol tolerating an arbitrary adversary structure $\mathbb{A}$ reduces to the existence of URMT protocols tolerating every two-sized subset of $\mathbb{A}$. Similar reduction can be found in [10].

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

Proof. Necessity is trivial. We give sufficiency proof here. We show how to construct a protocol for an adversary structure $\mathcal{A}$ of size $n > 2$ from protocols for adversary structures of smaller size. Using this technique, starting from protocols for adversary structures of size 2, we would be able to construct a protocol for an adversary structure of arbitrary size inductively.

Consider $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$, three $\lceil \frac{2\mathcal{A}}{3} \rceil$-sized subsets of $\mathcal{A}$ such that each element of $\mathcal{A}$ occurs in at least two of the three sets. For $1 \leq i \leq 3$, let $\Pi_{\mathcal{A}_i}$ be the URMT protocol tolerating $\mathcal{A}_i$. Let f be any field element **s** intends to send to **r**. Now, the protocol $\Pi_{\mathcal{A}}$ tolerating $\mathcal{A}$ is simply:

- Run the protocols $\Pi_{\mathcal{A}_1}$, $\Pi_{\mathcal{A}_2}$ and $\Pi_{\mathcal{A}_3}$ in the network, each one on the field element f.
- Receiver **r** outputs the majority of the outcome of these protocols.

Every $B \in \mathcal{A}$ is present in at least two subsets, hence the corresponding two protocols succeed with probability at least $(1 - \delta)$ each. Since **r** outputs majority of the outcomes, its output is correct when both these protocols terminate with the correct output which happens with at least $(1 - \delta)^2$ probability. Thus $\Pi_{\mathcal{A}}$ is an $(\mathcal{A}, 2\delta - \delta^2)$-URMT protocol which can be repeated sufficient number of times to achieve $(\mathcal{A}, \delta)$-URMT. $\square$

We now describe the conditions under which $(\mathcal{A}, \delta)$-URMT exists in a network G where $\mathcal{A} = \{B_1, B_2\}$ is a two-sized adversary structure.

5.1 Sufficiency

We remark that in this section we focus on characterization - we wish to capture all digraphs over which URMT is possible. To this end, we give 'a' protocol which can be run on any network over which some protocol for URMT exists. Hence, our protocol may not be the most efficient for every possible network. In fact, graphs can be constructed over which it may be impractical to run the protocols we describe here. We leave the task of designing efficient protocols or proving lower bounds as an interesting (and perhaps challenging) open problem. [2]

We first show how to construct a set Y iteratively such that if a node $u \in Y$, u can do URMT$_F$ to $\mathbf{r}$. Initialize $Y = \{\mathbf{r}\}$. Nodes are added to Y till there are no more nodes to add. An honest node $u \notin Y$ can be added to Y if one of the following hold:

1. $\exists\, v \in Y$ s.t. there is an honest path from u to v
2. Else, $\exists\, a \in Y$ s.t. there exists a path C avoiding B_2 from u to a and one of the following hold:
   (a) $\exists\, b \in Y$ s.t. $(b, u) \in E$ and $N(b) \cap B_1 = \phi$.
   (b) $\exists\, w \in B_2$ and $\exists\, b \in Y$ s.t. $(w, b), (w, u) \in E$ and $N(w) \cap B_1 = \phi$.
3. (a) and (b) are obtained by replacing B_1 with B_2 and vice versa in the construction 2(a) and 2(b) respectively.

The two constructions 2(a) and 2(b) are illustrated in Figure 2. Note that we only add honest nodes to $Y(\mathbf{r})$. Applying the constructions to digraph G_e in Figure 1, we can add nodes to Y in the following order: nodes q, u_4, y and u_3 are added first through Const 1, node u_2 is then added through Const 2(b), p is added through Const 2(a), u_1 is added through Const 1, x is added through Const 3(b), and finally $\mathbf{s}$ is added through Const 3(b).

In the following lemma, we show that no matter in what order nodes are added to Y, we always get the same result.

Lemma 2. *The set Y, constructed as described above, is well-defined, i.e., it has finite number of nodes and is unique.*

Proof. Finiteness is easy to see. To show uniqueness, consider two different sequences S_1 and S_2 in which nodes are added to Y; $S_1 = (x_1, x_2, \ldots, x_{n_1})$, $S_2 = (y_1, y_2, \ldots, y_{n_2})$ where $n_1, n_2 < |V|$ and each $x_i, y_j \in V$. Let k be the first index at which the two sequences differ, that is, $x_k \neq y_k$. Addition of x_k could have used only the nodes that were in S_1 already (i.e. $x_1, x_2, \ldots, x_{k-1}$), and certain paths in the network G (and/or a certain node in B_1 or B_2) which together satisfied some property. Since till index $k-1$ both the sequences are same and the network being used is also same, x_k would eventually appear at some index $k' > k$ in S_2. We can swap the elements present at k and k' in S_2 to obtain another sequence S_2'. Though S_2' contains same elements as in S_2, it matches S_1 upto first k locations, or, one more location as compared to S_2.

[2] We point out that efficient protocols for every undirected neighbour network (or every directed unicast network) over which URMT is possible are not known (for example, see [8, 10]).

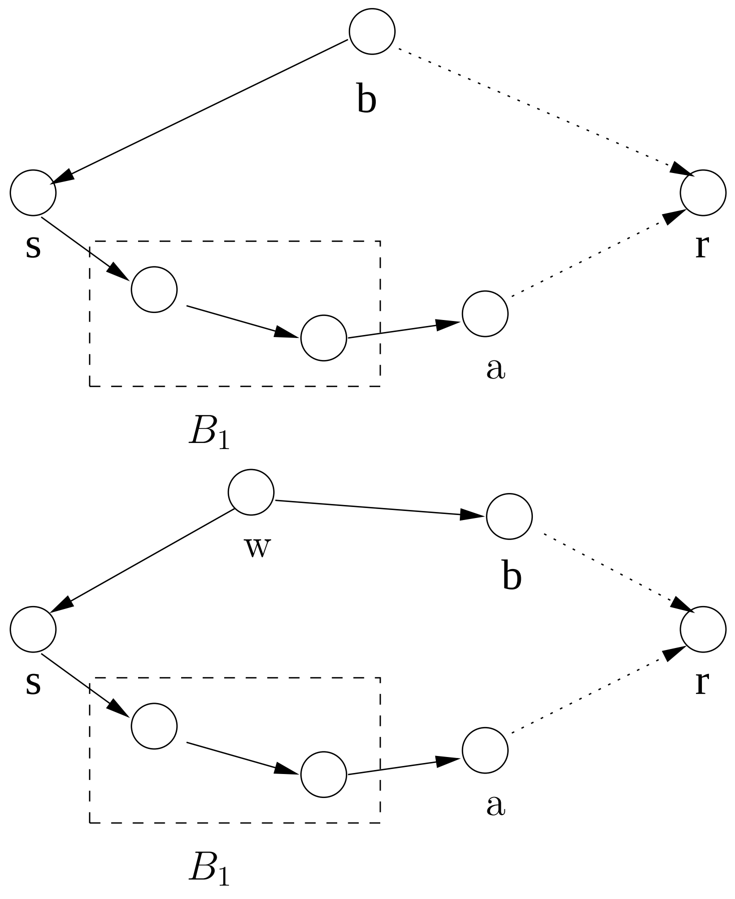

Figure 2: Constructions 2(a) and 2(b). Dotted edges between two nodes are *URMT$_F$ edges*, they may not be actually present in the graph. Box around nodes indicates that these nodes may belong to the set B_1.

The above procedure can be repeated sufficient number of times to obtain a sequence S_2^{final} from S_2 containing same elements as in S_2 but which exactly matches S_1 at all locations. Hence Y obtained through two different ways of adding nodes is same. $\square$

We now prove that every node $u \in Y$ can do URMT$_F$ to $\mathbf{r}$.

Lemma 3. *If $u \in Y$, there exists a protocol Γ_u for $(\{B_1, B_2\}, \frac{1}{|\mathbb{F}|})$-URMT$_F$ from u to $\boldsymbol{r}$.*

Proof. Let $\kappa = \frac{1}{|\mathbb{F}|}$. We give a proof by induction on the iteration at which a node is added. The protocol Γ_u which we construct inductively here satisfies the following two important properties: (a) If the corrupt nodes do not deviate from the protocol, $\mathbf{r}$ outputs the correct message with probability 1. (b) Otherwise, $\mathbf{r}$ knows the identity of the corrupt set with at least $1-\kappa$ probability (where the probabilities are taken over the random inputs of all honest players and the random inputs of the adversary). It is easy to see that a protocol satisfying these properties is a URMT$_F$ protocol of desired error probability κ. Also, due to these stronger properties, as we will see, the error probability remains bounded as we do induction.

An important remark is in order. Observe that in the constructions described above, a corrupt node can appear at two places only - in the path C or as node w. The protocol Γ_u we describe below uses node w to distribute keys and the path C to send authenticated messages. Hence, the adversary can disrupt the protocol in the following two ways only: (i) It can corrupt some nodes along path C and change the authenticate message tuple being sent through the path, (ii) It can corrupt the node w and distribute keys improperly.

The proof by induction goes as follows. The first node added to Y is $\mathbf{r}$ itself. Since, any node can send a message reliably to itself, protocol $\Gamma_{\mathbf{r}}$ is trivial. We now assume that $n-1$ nodes $z_1, z_2, \dots z_{n-1}$ have been added to Y; and for every $1 \leq i \leq n-1$, a protocol Γ_{z_i} satisfying the two properties described above exists for node z_i. Let z be the node added at the nth iteration through one of the constructions described above. Let $f \in \mathbb{F}$ be the message it intends to send. If z has been added through the first construction, it is straightforward to construct the protocol Γ_z – z sends f reliably along the honest path to v; now, Γ_v is run in the network with message f. When z is added through construction 2(b), the protocol Γ_z proceeds in the sequence of steps described below. A simpler version of this protocol works for construction 2(a), hence we do not discuss it separately here. The third construction can be dealt with similarly.

- Node $w \in B_2$ selects a random pair of keys $K_1, K_2 \in \mathbb{F}$ and sends it to b. Another neighbour of w - node z - receives the same message from w as b does (nodes are part of a neighbour network).

- If b does not receive two field elements, an instance of protocol Γ_b, say Γ_b^{fault}, is run as a sub-protocol in the network on message α_0. Otherwise, Γ_b^{fault} is run on message α_1. (Here α_0 and α_1 are two fixed distinct elements of the field.)

- Let z receive elements K_1', K_2'. If it does not, it chooses two elements at random. It sends $\chi(f; K_1', K_2')$ to a along C.

- If a receives a malformed message, it chooses two elements $f_1, f_2 \in \mathbb{F}$ at random. Otherwise, when a receives a well-formed message consisting of two field elements, let f_1, f_2 represent those two elements. Two instances of the protocol Γ_a – say Γ_a^1 and Γ_a^2 – are run as sub-protocols in the network on messages f_1 and f_2 respectively.

- After the first step, if b received two elements K_1 and K_2 from w, two instances of the protocol Γ_b, call them Γ_b^1 and Γ_b^2, are run as sub-protocols in the network *now* on messages K_1 and K_2 respectively. [3]

- *Reconstruction at* $\mathbf{r}$*:*
  - If the outcome of Γ_b^{fault} is α_0, $\mathbf{r}$ outputs 'B_2 is corrupt' and halts.
  - Else, $\mathbf{r}$ waits for the sub-protocols Γ_a^1, Γ_a^2, Γ_b^1 and Γ_b^2 to terminate in that order. If the outcome of a sub-protocol is 'B_1 is corrupt' or 'B_2 is corrupt', $\mathbf{r}$ outputs the same and halts, otherwise it waits for the next sub-protocol to terminate.
    Let $\mathbf{r}$ recover f_1^r, f_2^r from Γ_a^1 and Γ_a^2, and K_1^r, K_2^r from Γ_b^1 and Γ_b^2. It checks whether $f_2^r \overset{?}{=} f_1^r \cdot K_1^r + K_2^r$. If the verification succeeds it outputs f_1^r, otherwise it outputs 'B_1 is corrupt'.

[3]Even when b does not receive required number of elements, it can choose two elements randomly and Γ_b^1 and Γ_b^2 can be run respectively on them. Although this is not necessary, it can be introduced to *homogenize* the overall protocol.

It is easy to see that if the corrupt nodes follow the protocol, $\mathbf{r}$ halts with message f. We now show how any deviation from the protocol (recall that this can happen in two different ways only) almost surely reveals adversary's identity.

First, consider the four sub-protocols Γ_a^1, Γ_a^2, Γ_b^1 and Γ_b^2. If the corrupt nodes deviate during the first sub-protocol, $\mathbf{r}$ would find out the corrupt set in at least $1-\kappa$ fraction of executions of this sub-protocol and halt. On the other hand, if the corrupt nodes do not deviate during the first sub-protocol but deviate during the second one, $\mathbf{r}$ would recover the correct message from the first sub-protocol, and in at least $1-\kappa$ fraction of executions of the second sub-protocol it would come to know the corrupt set. If we continue to argue like this, we can say that if the corrupt set of nodes deviate during any of the four sub-protocols, $\mathbf{r}$ would know the corrupt set in at least $1-\kappa$ fraction of executions of the protocol Γ_z.

Now, we assume that the corrupted nodes do not deviate during any of the four sub-protocols. This implies that $\mathbf{r}$ would correctly recover the messages sent through these sub-protocols. Suppose set B_2 is corrupt and $w \in B_2$ does not distribute keys properly in first step. In this case, b would send the message α_0 to $\mathbf{r}$ through the sub-protocol Γ_b^{fault}. Node $\mathbf{r}$ would recover α_0 with at least $1-\kappa$ probability, knows that B_2 is corrupt and halts. On the other hand, when B_1 is corrupt, adversary may modify the pair $\chi(f; K_1, K_2)$ sent through path C in third step. Since w has no neighbour in B_1, adversary has no knowledge about the keys K_1, K_2. Hence, any modification of the pair leads to the failure of verification in Step 2 of Reconstruction at $\mathbf{r}$ with atleast $1-\kappa$ probability (as discussed in the last part of Section 3). The corrupt nodes on path C may also forward a malformed message to a. In this case a himself generates a random pair of elements which again fail verification with high probability. □

We are now ready to state our main theorem of the paper.

Theorem 4. *In a directed neighbour network $\mathcal{G} = (V, \mathcal{E})$, $(\{B_1, B_2\}, \delta)$-URMT from $\mathbf{s}$ to $\mathbf{r}$ is possible if and only if there exist two paths from $\mathbf{s}$ to $\mathbf{r}$, one avoiding B_1, another avoiding B_2, and $\mathbf{s} \in Y$.*

Proof. Sufficiency: For $i \in \{1, 2\}$, let C_i be the path avoiding B_i. Sender $\mathbf{s}$ sends the message f to $\mathbf{r}$ through the two paths C_1 and C_2. Next, Γ_s (constructed in the previous lemma) is run in the network on message f. With at least $1 - \frac{1}{|\mathbb{F}|}$ probability, $\mathbf{r}$ receives the correct message or knows that the set B_i is corrupt. In the latter case, it outputs the message received along the path C_i.

We prove necessity in the following section. □

6 Necessity

In this section we show that if a neighbour network G does not satisfy the conditions mentioned in Theorem 4 then $(\{B_1, B_2\}, \delta)$-URMT is impossible from $\mathbf{s}$ to $\mathbf{r}$ in G. It is easy to see that $\mathbf{s}$ must have a path to $\mathbf{r}$ avoiding B_1, otherwise B_1 would fail-stop and no communication from $\mathbf{s}$ to $\mathbf{r}$ would be possible. Similarly, the existence of another path avoiding B_2 is necessary. We now assume that $\mathbf{s} \notin Y(\mathbf{r})$. We also assume that the sets B_1 and B_2 are disjoint. If not, an additional part of the adversary strategy would be to always fail-stop nodes in $B_1 \cap B_2$ no matter which among the two sets B_1, B_2 is chosen to be corrupted.

We divide the set of nodes in B_1 into two disjoint sets. Let $B_1^Y = \{u \in B_1 : \exists$ a path from u to a node $v \in Y$ avoiding $B_2\}$. Let $B_1^{Y'} = B_1 \setminus B_1^Y$. Similarly we can divide the set B_2 into two disjoint sets. We also define a set B_1^d of nodes in B_1^Y who have a direct edge to a node in

Y, i.e., $B_1^d = \{u \in B_1^Y : \exists\, v \in Y \text{ s.t. } (u,v) \in E\}$. Further, let $B_1^{d'} = B_1^Y \setminus B_1^d$. Symmetrically, B_2^d and $B_2^{d'}$ are also defined.

We next divide the set of nodes $X = V \setminus (Y \cup B_1 \cup B_2)$ into four disjoint sets. Let $X_{1/2} = \{u \in X : u$ has a path to a node $v \in Y$ avoiding either B_1 or $B_2\}$. Let $X'_{1/2} = X \setminus X_{1/2}$. We further divide the set $X_{1/2}$, into three disjoint sets. Let $X_{1\&2} = \{u \in X_{1/2} : \exists\, v_1, v_2 \in Y$ s.t. u has a path to v_1 avoiding B_1 and a path to v_2 avoiding $B_2\}$. Let $X_1 = \{u \in X_{1/2} : \nexists\, v \in Y$ s.t. u has a path to v avoiding $B_1\}$. Likewise X_2 is also defined. It is easy to see that $X_{1\&2}, X_1$ and X_2 are disjoint sets. Also, $\mathbf{s} \in X_{1\&2}$.

In the following, when we say that there cannot be an edge from a set Z_1 to a set Z_2, we actually mean that there does not exist a pair of nodes $u_1 \in Z_1$ and $u_2 \in Z_2$ s.t. $(u_1, u_2) \in E$. We first prove some simple Lemmas regarding the connectivity among the sets we have just defined, see Figure **??** in Appendix **??**.

Lemma 5. *(a) If $(u,v) \in E$ and $u \in B_1^{Y'}$ then $v \in B_1^{Y'} \cup B_2 \cup X_2 \cup X'_{1/2}$.*
(b) If $(u,v) \in E$ and $u \in B_2^{Y'}$ then $v \in B_2^{Y'} \cup B_1 \cup X_1 \cup X'_{1/2}$.

Proof. (a) According to definition, a node u belongs to $B_1^{Y'}$ if it does not have a path avoiding B_2 to Y. Hence, u cannot have an edge to Y or B_1^Y. Also, both $X_{1\&2}$ and X_1 have paths to Y avoiding B_2. Hence, there cannot be an edge from $B_1^{Y'}$ to either of them. Finally, $V \setminus (Y \cup B_1^Y \cup X_{1\&2} \cup X_1) = B_1^{Y'} \cup B_2 \cup X_2 \cup X'_{1/2}$. (b) can be similarly proved. □

Note that there cannot be an edge from X to Y otherwise a node in X would move to Y by Construction 1 (described in 5.1).

Lemma 6. *(a) If $(u,v) \in E$ and $u \in X_1$ then $v \in X_1 \cup B_1 \cup B_2^{Y'} \cup X'_{1/2}$.*
(b) If $(u,v) \in E$ and $u \in X_2$ then $v \in X_2 \cup B_2 \cup B_1^{Y'} \cup X'_{1/2}$.

Proof. (a) As $X_1 \subseteq X$, we know that it cannot have an edge to Y. Since a node u in X_1 must not have a path avoiding B_1 to Y, u cannot have an edge to B_2^Y. Also, both $X_{1\&2}$ and X_2 have paths to Y avoiding B_1. Hence, there cannot be an edge from X_1 to either of them. Finally, $V \setminus (Y \cup B_2^Y \cup X_{1\&2} \cup X_2) = X_1 \cup B_1 \cup B_2^{Y'} \cup X'_{1/2}$. (b) can be similarly proved. □

Lemma 7. *If $(u,v) \in E$ and $u \in X'_{1/2}$ then $v \in X'_{1/2} \cup B_1^{Y'} \cup B_2^{Y'}$.*

Proof. As $X'_{1/2} \subseteq X$, we know that it cannot have an edge to Y. Since $X'_{1/2}$ contains nodes from whom all paths to Y pass through both B_1 and B_2, it cannot have an edge to either $X_{1/2}$ or B_1^Y or B_2^Y. □

Lemma 8. *(a) If $u \in Y \cup B_2^d$ and $N(u) \cap (X_{1\&2} \cup X_1) \neq \phi$ then $N(u) \cap B_1 \neq \phi$.*
(b) If $u \in Y \cup B_1^d$ and $N(u) \cap (X_{1\&2} \cup X_2) \neq \phi$ then $N(u) \cap B_2 \neq \phi$.

Proof. (a) We know that every node in $X_{1,2}$ and X_1 has at least one path to Y avoiding B_2. If a node $u \in Y$ (resp. $u \in B_2^d$) has a neighbour v in $X_{1\&2} \cup X_1$ but no neighbour in B_1, v can do URMT$_F$ to $\mathbf{r}$ by Construction 2a (resp. Construction 2b) as described in 5.1. Since $v \notin Y$, this is not allowed. (b) can be proved similarly. □

Before proceeding further, we define few more sets to make our presentation compact. Define:

- $V_1 = X_1 \cup X_{1\&2}$ and $V_2 = X_2 \cup X_{1\&2}$
- $E_1^1 = \{(u,v) \in E : u, v \in V_1\}$ and similarly E_2^1

- $E_1^2 = \{(u,v) \in E : u \in V_1 \text{ and } v \in B_1^Y \text{ OR } u \in B_1^Y \text{ and } v \in V_1\}$, and similarly E_2^2
- $E_1^3 = \{(u,v) \in E : u, v \in V_1 \text{ and } v \in B_2^d \cup Y\}$, and similarly E_2^3
- $E_1 = E_1^1 \cup E_1^2 \cup E_1^3$, and similarly E_2
- $E_{perm} = \{(u,v) \in E : u, v \in B_1^d \cup B_2^d \cup Y \text{ OR } u \in B_1^{d'} \cup B_2^{d'} \text{ and } v \in B_1^d \cup B_2^d\}$

We now describe the adversary strategy (ζ_1, ζ_2). When adversary corrupts the set B_1 during an execution of a protocol, it follows the strategy ζ_1. It fails-stops all nodes in $B_1^{Y'}$ at the start of the execution. During the entire execution, it drops all messages received from nodes in $X \cup (B_2 \setminus B_2^d)$. Moreover, $B_1^{d'}$ doesn't send any message on its outgoing edges throughout the execution. However, the most important component of the adversary strategy is simulation. To make simulated sets easily distinguishable from *actual* sets, we use curly alphabets for the former and normal alphabets for the latter.

Adversary simulates a sub-graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ of the graph G. Here $\mathcal{V}_1$ is the union of sets $\mathcal{X}_1$ and $\mathcal{X}_{1\&2}$ which represent the simulated copies of nodes in X_1 and $X_{1\&2}$ respectively. Also, $\mathcal{E}_1$ is the union of sets $\mathcal{E}_1^1$, $\mathcal{E}_1^2$ and $\mathcal{E}_1^3$ which represent the simulated copies of edges in E_1^1, E_1^2 and E_1^3 respectively. Since the adversary controls all the nodes in set B_1, simulation of edges $\mathcal{E}_1^2$ is straightforward. Moreover, from Lemma 8 we know that if a node $z \in B_2^d \cup Y$ has a neighbour in $X_1 \cup X_{1\&2}$, it also has a neighbour in B_1. Hence, the messages which nodes in $B_2^d \cup Y$ send to nodes in $X_1 \cup X_{1\&2}$ during the execution are also available to the adversary leading to a successful simulation of edge set $\mathcal{E}_1^3$.

In the following, we may not explicitly mention messages exchanged between nodes of a set when it is clear from the context that they would. The simulated sender $s_1 \in \mathcal{X}_{1\&2}$ (say) is given a message f_1 as input at the beginning of the execution. During each round of the execution, simulated nodes in $\mathcal{V}_1$ receive messages from each other through the simulated edges $\mathcal{E}_1^1$, from B_1^Y through the simulated edges in $\mathcal{E}_1^2$ and from $B_2^d \cup Y$ through the simulated edges $\mathcal{E}_1^3$. Nodes in B_1^Y receive messages from $\mathcal{V}_1$ through the simulated edges in $\mathcal{E}_1^2$ and from nodes in $B_2^d \cup Y$. The simulated nodes $\mathcal{V}_1$ and the nodes B_1^Y run their part of the protocol with these input messages. They exchange output messages among each other through the simulated edges in $\mathcal{E}_1^2$. While B_1^d sends output messages on all its *actual* outgoing edges, $B_1^{d'}$ does not send any message on any of its *actual* outgoing edges. [4]

This completes the description of strategy ζ_1. Analogously, we can describe the other part ζ_2. Among other things, when following ζ_2, adversary simulates the sub-graph $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ where, in short, $\mathcal{V}_2$ represents the simulated copies of nodes in V_2 and $\mathcal{E}_2$ represents the simulated copies of edges in E_2.

The following Lemma shows how the adversary strategy (ζ_1, ζ_2) *cuts-off* some nodes from the graph.

Lemma 9. *In an execution when adversary corrupts B_1, no messages from nodes in $B_2^{Y'} \cup X_1 \cup X'_{1/2}$ can reach Y. Similarly, in an execution when adversary corrupts B_2, no messages from nodes in $B_1^{Y'} \cup X_2 \cup X'_{1/2}$ can reach Y.*

Proof. To prove this, we find out to what other sets a certain set can send messages to. From Lemmas 5 6 7, we observe the following: $B_2^{Y'}$ can only send messages to B_1, X_1 and $X'_{1/2}$; X_1 can only send messages to B_1, $B_2^{Y'}$ and $X'_{1/2}$; $X'_{1/2}$ can only send messages to $B_1^{Y'}$ and $B_2^{Y'}$.

[4] An important thing to note is that since we are dealing with a neighbour network, a node in B_1^Y cannot choose to send messages to *some* of its neighbours in other sets and not to others.

As $B_2^{Y'} \cup X_1 \cup X'_{1/2} \subseteq X \cup (B_2 \setminus B_2^d)$, B_1 ignores messages from it. Henceforth, the nodes in $B_2^{Y'} \cup X_1 \cup X'_{1/2}$ form a *closed system* which can send messages only among themselves. None of the messages originating from them is received by B_2^Y. Since nodes in Y receive messages from B_1^Y and B_2^Y, in an execution where B_1 is corrupt, no messages from nodes in $B_2^{Y'} \cup X_1 \cup X'_{1/2}$ can reach Y. On the same lines, the other part can be proved. □

Omitting few details here, we can say that w.r.t to the set Y which contains **r**, when adversary corrupts B_1, the network effectively reduces to the node sets $\mathcal{V}_1$, B_1^Y, V_2, B_2^Y, Y and the edge sets $\mathcal{E}_1$, E_2, E_{perm} (see Figure 3). Similarly, when adversary corrupts B_2, it reduces to V_1, B_1^Y, $\mathcal{V}_2$, B_2^Y, Y and E_1, $\mathcal{E}_2$, E_{perm}. This forms the basis of indistinguishability of views at **r**.

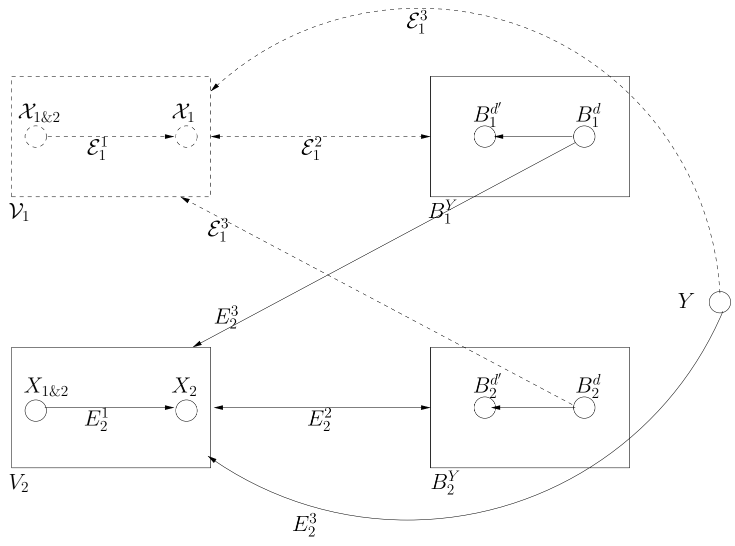

Figure 3: A pictorial view of adversary strategy ζ_1. Dotted circles represent simulated nodes, dotted lines/arcs represent simulated edges. The edge set E_{perm} is not complete shown for sake of clarity.

We now show that under the adversary strategy (ζ_1, ζ_2) no $(\mathcal{A}, \delta)$-URMT protocol can exist. On the contrary, assume there exists a protocol Π for $(\mathcal{A}, \delta)$-URMT from **s** to **r**. Consider an execution E_1 of Π: Sender **s** chooses to send f_2. Adversary corrupts B_1 and follows strategy ζ_1. (Recall that the simulated sender gets f_1 as input.) Let coin tosses of nodes in V_2 and B_2^Y be collectively denoted by r_{V_2} and $r_{B_2^Y}$ respectively. Similarly, let coin tosses of sets $\mathcal{V}_1$ and B_1^Y under the control of adversary be denoted by $r_{\mathcal{V}_1}$ and $r_{B_1^Y}$ respectively. Coin tosses of nodes in Y are r_Y. Note that, by Lemma 9, we need not take nodes in other sets into consideration.

Consider another execution E_2 of Π: Sender **s** chooses to send f_1. Adversary corrupts B_2 and follows strategy ζ_2. (Recall that the simulated sender gets f_2 as input.) Let coin tosses of nodes in V_1 and B_1^Y be collectively denoted by r_{V_1} and $r_{B_1^Y}$ respectively. Similarly, let coin tosses of sets $\mathcal{V}_2$ and B_2^Y under the control of adversary be denoted by $r_{\mathcal{V}_2}$ and $r_{B_2^Y}$ respectively. Coin tosses of nodes in Y are r_Y.

We can see that the view at Y, and hence at **r**, in both the executions E_1 and E_2 is same. Therefore **r**'s output will be same in both of them. In general, for every execution in which **s** sends f_2, there exists an execution in which **s** chooses to send f_1 and the view at **r** is same, and vice versa. Since Π is a URMT protocol, when **s** chooses to send message f_2, for at least $1 - \delta$

fraction of executions receiver **r** must output f_2. Now, **r** would again output f_2 in at least $1 - \delta$ fraction of executions in which **s** chooses to send message f_1. Hence, Π is not a valid URMT protocol. This completes our necessity proof.

7 Conclusion

In this work, we have characterized directed neighbour networks over which URMT tolerating non-threshold Byzantine adversary is possible. However, Byzantine faults are not the only faults present in a network. There may be *simpler* faults like fail-stop, omission, etc. A network may tolerate only upto a certain number of Byzantine faults, but it may additionally tolerate a few *simpler* faults. This is captured via a *mixed* adversary (see [10]). It would be interesting to generalize our results in neighbour networks to the case of *mixed* adversary. Our work can also be extended to a more general network model like directed hypergraphs, which is a better model in a variety of situations. Though characterization exists for this case [9], it is very complicated and is not supplemented by a rigorous necessity proof. As discussed earlier, the problem of designing efficient protocols for directed neighbour networks also remains open.

References

[1] A Choudhary, A Patra, B V Ashwinkumar, K Srinathan, and C P 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.

[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] 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.

[4] M Franklin and M Yung. Secure Hypergraphs: Privacy from Partial Broadcast. In *Proceedings of 27th Symposium on Theory of Computing (STOC)*, pages 36–44. ACM Press, 1995.

[5] 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.

[6] L Lamport, R Shostak, and M Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[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] J Renault and T Tomala. Probabilistic reliability and privacy of communication using multicast in general neighbor networks. *Journal of Cryptology*, 21(2):250–279, April 2008.

[9] K Srinathan, A Patra, A Choudhary, and C P Rangan. Unconditionally reliable message transmission in directed hypergraphs. In *Cryptology and Network Security*, volume 5339 of *Lecture Notes in Computer Science*, pages 285–303. Springer Berlin / Heidelberg, 2008.

[10] K Srinathan and C P 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.

[11] Y Wang. Robust key establishment in sensor networks. *SIGMOD Rec.*, 33(1):14–19, 2004.