Unconditionally Reliable Message Transmission in Directed Networks

Bhavani Shankar* shankar@research.iiit.ac.in
Prasant Gopal* prasant_a@students.iiit.ac.in
Kannan Srinathan* srinathan@iiit.ac.in
C. Pandu Rangan† rangan@iitm.ernet.in

Abstract

In the *unconditionally reliable message transmission* (URMT) problem, two non-faulty players, the *sender* **S** and the *receiver* **R** are part of a synchronous network modeled as a directed graph. **S** has a message that he wishes to send to **R**; the challenge is to design a protocol such that after exchanging messages as per the protocol, the receiver **R** should *correctly* obtain **S**'s message with arbitrarily small error probability δ, in spite of the influence of a Byzantine adversary that may actively corrupt up to t nodes in the network (we denote such a URMT protocol as $(t,(1-\delta))$-reliable). While it is known that $(2t+1)$ vertex disjoint directed paths from **S** to **R** are necessary and sufficient for $(t,1)$-reliable URMT (that is with zero error probability), we prove that a strictly weaker condition, which we define and denote as $(2t,t)$-special-connectivity, together with just $(t+1)$ vertex disjoint directed paths from **S** to **R**, is necessary and sufficient for $(t,(1-\delta))$-reliable URMT with arbitrarily small (but non-zero) error probability, δ. Thus, we demonstrate the power of randomization in the context of reliable message transmission. In fact, for any positive integer $k>0$, we show that there always exists a digraph G_k such that $(k,1)$-reliable URMT is *impossible* over G_k whereas there exists a $(2k,(1-\delta))$-reliable URMT protocol, $\delta>0$ in G_k.

In a digraph G on which $(t,(1-\delta))$-reliable URMT is possible, an edge is called *critical* if the deletion of that edge renders $(t,(1-\delta))$-reliable URMT impossible. We give an example of a digraph G on n vertices such that G has $\Omega(n^2)$ *critical* edges. This is quite baffling since no such graph exists for the case of perfect reliable message transmission (or equivalently $(t,1)$-reliable URMT) or when the underlying graph is undirected. Such is the anomalous behavior of URMT protocols (when "randomness meet directedness") that it makes it extremely hard to design efficient protocols over arbitrary digraphs. However, if URMT is possible between every pair of vertices in the network, then we present *efficient* protocols for the same.

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

†Department of Computer Science and Engineering, Indian Institute of Technology, Madras, Chennai, 600036, India.

1 Introduction

Consider a synchronous network (modeled as a directed graph) denoted by $\mathcal{N}=(\mathbb{P},\mathcal{E})$, where $\mathbb{P}$ is the set of nodes and $\mathcal{E}\subseteq\mathbb{P}\times\mathbb{P}$ is the set of (directed) links. In the *unconditionally reliable message transmission* (URMT) problem over $\mathcal{N}$, the sender $\mathbf{S}\in\mathbb{P}$ wishes to send a message m to the receiver $\mathbf{R}\in\mathbb{P}$, in a robust manner such that the message is correctly received by **R** with a very high probability, in spite of presence of up to t Byzantine-faulty nodes in $\mathcal{N}$. Specifically, a URMT protocol is said to be $(t,1-\delta)$-reliable if **R** outputs the correct message with a probability of at least $(1-\delta)$.

A protocol for URMT is one of the fundamental primitives used by almost all fault-tolerant distributed algorithms since without reliable communication little that is truly collaborative is possible. In fact, several popular fault-tolerant distributed algorithms, like (randomized) Byzantine agreement etcetera, assume that the underlying network is a complete graph, thereby implicitly assuming the existence of a URMT protocol that can simulate a complete graph overlaid in the actual underlying network (for the actual connectivity is seldom complete in practice). Notwithstanding its applications in distributed computing, the problem of URMT is nevertheless, in principle, interesting and challenging in own right.

We use a digraph to capture the underlying communication network. We stress that in practice not every communication channel admits bi-directional communication (for instance, a base-station may communicate to even a far-off hand-held device but the other way round is not possible) and hence the digraph model is practically well-motivated. Furthermore, directed graphs are a strict generalization and hence the results of this paper are adaptable/applicable to the undirected graph model too. Curiously enough, there is yet another reason for studying protocols across directed graphs. In [4], it is proved that across an undirected graph G influenced by an adversary $\mathcal{A}$,

an r-round URMT protocol exists if and only if there exists a URMT protocol in a *digraph* H influenced by a "related" adversary $\mathcal{B}$ wherein H and $\mathcal{B}$ are easily computed given $G, \mathcal{A}$, and r. Since one could perform a binary-search for the optimal r, it is clear that characterizing the *possibility* of URMT in digraphs amounts to algorithmically characterizing the *round-optimality* of URMT over undirected graphs! Thus, even if one is hesitant in studying directed graphs per say, he would now like to study the same as a part of his toolkit needed for the design of optimal URMT protocols in undirected graphs!

The problem of perfectly reliable message transmission (PRMT) was first studied in [2]. It is shown in [2] that across a synchronous undirected network/graph under the influence of a Byzantine adversary that may corrupt up to any t nodes in the network, $(t, 1)$-reliable URMT from **S** to **R** is possible if and only if there exist $(2t+1)$ vertex disjoint paths from **S** to **R** in the network. Subsequently, in [3], the above result was extended for any $\delta > \frac{1}{2}$; in other words, it is proved that (t, δ)-reliable URMT from **S** to **R** is possible if and only if there exist $(2t + 1)$ vertex disjoint paths from **S** to **R** in the network. Furthermore, it follows from the results of [1] that across a directed network, $(t, 1)$-reliable URMT from **S** to **R** is possible if and only if there exist $(2t + 1)$ vertex disjoint *directed* paths from **S** to **R** in the network. We now ask *what is the necessary and sufficient condition for the possibility of* (t, δ)*-reliable URMT,* $\delta > \frac{1}{2}$*, across a synchronous directed network influenced by a Byzantine adversary that can corrupt up to any* t *nodes in the network?* Of course, we expect it to be something in the lines of: "across a directed network, (t, δ)-reliable URMT, $\delta > \frac{1}{2}$, from **S** to **R** is possible if and only if there exist $(2t + 1)$ vertex disjoint *directed* paths from **S** to **R** in the network." Surprisingly, we show that this is *far from true*! We present, in Section 4, a simple yet strange characterization for δ-reliable URMT in directed graphs tolerating *threshold* Byzantine adversary that corrupts up t nodes in the network. Note that such an anomalous behavior of reliable communication when "randomization meets directedness" is already known in the literature for the case of generalized non-threshold adversaries [4]. However, unlike our characterization, due to the generality of non-threshold adversaries the characterization in [4] is very complex.

2 Model and Definitions

The network is modeled as a directed graph $\mathcal{N} = (\mathbb{P}, \mathcal{E})$ where $\mathbb{P}$ is the set of vertices and $\mathcal{E}$ denotes the set of arcs/edges in the directed graph. The system is assumed to be synchronous, that is, the protocol is executed in a sequence of *rounds* wherein in each round, a player can perform some local computation, send new messages to his out-neighbors, receive the messages sent in that round by his in-neighbors (and if necessary perform some more local computation), in that order.

In the graph, we assume that the channels are *secure*. In other words, if $(u, v) \in \mathcal{E}$ then what it means is that *the player* u *can securely send a message to player* v *in one round*. During the execution, the adversary may be corrupted up to any t players/nodes. The adversary may completely control all the corrupted players and make them behave in an arbitrary way.

We now define what we mean by a URMT protocol. We assume that the sender **S** and the receiver **R** are honest.

Definition 2.1. (URMT protocol) *Let* $\mathcal{N} = (\mathbb{P}, \mathcal{E})$ *be a network under the influence of a Byzantine adversary that may corrupt up to any* t *nodes. We say that a protocol for transmitting a message from* **S** *to* **R** *is* $(t, 1-\delta)$*-reliable if for any valid adversary strategy, the probability that* **R** *outputs* $\mathbf{m}$ *given that* **S** *has sent* $\mathbf{m}$*, is at least* $(1-\delta)$ *where the probability is over the random inputs of all the players and random inputs of the adversary.*

Definition 2.2. (Strong Path) *A sequence of vertices* $v_1, v_2, v_3, \ldots, v_k$ *is said to be a strong path from* v_1 *to* v_k *in the network* $\mathcal{N} = (\mathbb{P}, \mathcal{E})$ *if for each* $1 \leq i < k$, $(v_i, v_{i+1}) \in \mathcal{E}$*. Furthermore, we assume that there vacuously exists a strong path from a node to itself.*

Definition 2.3. (t-(**S,R**)-strong-connectivity) *A digraph is said to be* t*-(***S,R***)-strong-connected if the graph is such that there exists at least* t *vertex disjoint strong paths from* **S** *to* **R**.

Definition 2.4. (Weak Path) *A sequence of vertices* $v_1, v_2, v_3, \ldots, v_k$ *is said to be a weak path from* v_1 *to* v_k *in the network* $\mathcal{N} = (\mathbb{P}, \mathcal{E})$ *if for each* $1 \leq i < k$, *either* $(v_i, v_{i+1}) \in \mathcal{E}$ *or* $(v_{i+1}, v_i) \in \mathcal{E}$.

We now define the notion of (t_1, t_2)-(**S,R**)-special-connectivity which will be directly used in our characterization theorems. Note that it is a strange definition markedly different from the standard expected definitions of connectivity. Also note that the definitions depend on who the receiver **R** is.

Definition 2.5. ((t_1, t_2)-(**S,R**)-special-connectivity) *A network* $\mathcal{N} = (\mathbb{P}, \mathcal{E})$ *is said to be* (t_1, t_2)*-(***S,R***)-special-connected if on the deletion of any set* D *of*

at most t_1 nodes (that is $|D| \leq t_1$) from $\mathcal{N}$ other than **S** *and* **R**, *there exists a weak path p in the rest of the network (namely, the sub graph induced by $\mathcal{N}$ on $(\mathbb{P} \setminus D)$) such that for every node w in the path p that has both its adjacent edges (along path p) directed inward towards w the following holds: it should* **not** *be possible to divide the set D into two sets D_1 and D_2, $|D_1| \leq t_2$, $|D_2| \leq t_2$, $D_1 \cup D_2 = D$, such that in the original network $\mathcal{N}$, both D_1 and D_2 are vertex cut-sets between w and* **R**, *that is, every strong path from w to* **R** *passes through nodes in both D_1 and D_2.*

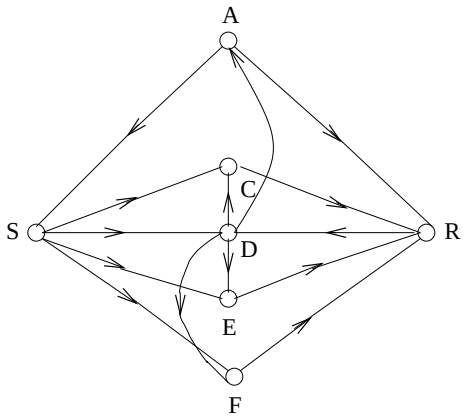

Figure 1: Example of a $(4, 2)$-(**S,R**)-special-connected graph.

3 Our Results

In Section 4 we present an elegant characterization for the possibility of URMT in directed networks influenced by (static) Byzantine adversary. Specifically, we prove that URMT is possible if and only if the network is both $(2t, t)$-(**S,R**)-special-connected and $(t + 1)$-(**S,R**)-strong-connected. From this result, it follows that there are several digraphs across which PRMT (or equivalently $(t, 1)$-URMT) is *impossible* whereas URMT (with a arbitrary small error probability) is *possible*. The power of randomization in the context of fault-tolerance in reliable communication is thus brought to the fore. However, we do not focus on the complexity of the designed URMT protocols; in fact, our naive constructions to prove the existence/possibility of URMT protocols invariably lead to super-polynomial complex solutions. We leave the design of efficient URMT protocols (whenever URMT is possible at all) as open. We remark that achieving polynomial complexity is quite challenging and actually we have not yet ruled out the possibility of a super-polynomial lower bound.

In Section 5, we show that for any positive integer $k > 0$, there always exists a digraph G_k such that $(k, 1)$-reliable URMT is *impossible* over G_k whereas there exists a $(2k, (1 - \delta))$-reliable URMT protocol, $\delta > 0$ in G_k. Furthermore, the achieved "gap", namely, by a factor of two is provably optimal.

In a digraph G on which $(t, (1-\delta))$-reliable URMT is possible, an edge is called *critical* if the deletion of that edge renders $(t, (1-\delta))$-reliable URMT impossible. In Section 6 we give an example of a digraph G on n vertices such that G has $\Omega(n^2)$ *critical* edges. This is quite baffling since no such graph exists for the case of PRMT (or equivalently $(t, 1)$-reliable URMT) when the underlying graph is undirected. Such anomalies are the evidences for our conjecture that the design of efficient URMT protocols for general graphs is highly non-trivial.

Next, in Section 7 we present *efficient* URMT protocols over a class of graphs, namely, ones in which URMT is possible between every pair of nodes.

4 Characterizing URMT in Directed Graphs

THEOREM 4.1. *$(t, (1-\delta))$-reliable URMT, for an arbitrarily small but positive δ, from* **S** *to* **R** *in the network (directed graph) $\mathcal{N} = (\mathbb{P}, \mathcal{E})$ tolerating a static Byzantine adversary characterized that may corrupt up to any t nodes, is possible if and only if the network $\mathcal{N}$ is $(2t, t)$-(**S,R**)-special-connected as well as $(t+1)$-(**S,R**)-strong-connected.*

Proof. Sufficiency: We now prove that if the network is $(2t, t)$-(**S,R**)-special-connected as well as $(t+1)$-(**S,R**)-strong-connected, then a protocol for $(t, (1-\delta))$-reliable URMT exists. The proof outline is as follows: it is clear that the adversary can corrupt no more that t nodes in the network; we design a protocol by assuming that the adversary *always* uses his full power and corrupts exactly t nodes. It is straightforward to see that such a protocol would also withstand an adversary that does not always corrupt t nodes.

Now, our modified adversary has $\binom{n}{t}$ options in front of him of which he may choose one — that is there are exactly $\binom{n}{t}$ distinct ways of corrupting exactly t nodes. Let us enumerate these options by writing down each of the $\binom{n}{t}$ distinct subsets of size t each, say, $\{B_1, B_2, B_3, \ldots, B_{\binom{n}{t}}\}$ where $B_i \subset \mathbb{P}$ and $|B_i| = t$.

First, we show how to design a "URMT sub-protocol" assuming that the adversary is allowed to choose only from *two* of the $\binom{n}{t}$ options that originally

existed. In other words, we are only concerned about an adversary that may corrupt the nodes in the set B_α or the set B_β, where $1 \leq \alpha, \beta \leq \binom{n}{t}$ and $\alpha \neq \beta$. Let us denote the resulting sub-protocol as $\Pi_{\alpha\beta}$. In the sequel, we show how to use all the sub-protocols $\Pi_{\alpha\beta}$ (there are clearly $\binom{\binom{n}{t}}{2}$ of them) to design a grand protocol Π that can be proved to be a $(t, (1-\delta))$-reliable URMT protocol.

Designing the sub-protocol $\Pi_{\alpha\beta}$: We know that the total number of nodes in the set $B_\alpha \cup B_\beta$ is at most $2t$. On the other hand, since the network is $(2t, t)$-(**S**,**R**)-special-connected, there must exist $(2t+1)$ vertex disjoint weak paths from **S** to **R**. Thus, at least one of these vertex disjoint paths is completely *honest*. Let p be that honest path. Now, we consider the following two cases in the design of $\Pi_{\alpha\beta}$:

Case 1: *The path p is such that there is no node w along p with both the adjacent edges directed in-ward towards w:* In such a case, it is apparent that the path p must contain a node y (which may be **S** or **R** too) such that p is the combination of the strong path from y to **S** and the strong path from y to **R**. Consider the following protocol: First y sends three random keys K_1, K_2 and K_3 (all elements of a finite field $\mathbb{F}$, and all computations in the sequel are performed over $\mathbb{F}$) to both **S** and **R** using the path p. Next, **S** sends a value ψ and a signature χ through *all* the $(t+1)$ strong paths to **R**, where $\psi = (M + K_1)$, $\chi = (K_2(M + K_1) + K_3)$ and M is the message that needs to be reliably transmitted. Now, **R** receives a value(ψ') and its signature(χ') based on the keys (Notice that, **R** has knowledge of K_1, K_2 and K_3 and hence can easily verify if $\chi' \stackrel{?}{=} K_2 * \psi' + K_3$). **R** reacts as follows: If the received value ψ' has a valid signature ($\chi' = K_2 * \psi' + K_3$), then **R** outputs $(\psi' - K_1)$; furthermore, among the $(t+1)$ received values, at least one of them is guaranteed to be valid.[1] Thus, **R**, with a high probability (namely $1 - \frac{1}{|\mathbb{F}|}$ here which can be made $(1-\delta)$ by suitably choosing $\mathbb{F}$) will output the correct message M.

Case 2: *The path p is such that there are $k > 0$ nodes $w_1, \ldots, w_k$ along p such that each w_i has both the adjacent edges directed in-ward towards itself, for all $1 \leq i \leq k$:* We will first consider the case when $k = 1$. By definition of $(2t, t)$-(**S**,**R**)-special-connectivity, there must exist a strong path Q from w_1 to **R** that does not pass through nodes in either the set B_α or the set B_β. Recall that p must contain a node y (which may be **R**) such that there is strong path from y to w_1 (along p) and there is a strong path from y to **R** (also along p). Consider the following protocol: First y sends three random keys K_1, K_2 and K_3 (all elements of a finite field $\mathbb{F}$, and all computations in the sequel are performed over $\mathbb{F}$) to both w_1 and **R** using the path p. Next, w_1 sends a value ψ and a signature χ through Q to **R**, where $\psi = (M+K_1)$, $\chi = (K_2(M+K_1)+K_3)$ and M is the message that needs to be reliably transmitted. Now, **R** receives a value(ψ') and its signature(χ') based on the keys along the path Q (Notice that, **R** has knowledge of K_1, K_2 and K_3 and hence can easily verify if $\chi' \stackrel{?}{=} K_2 * \psi' + K_3$). **R** reacts as follows: if the received value ψ' has a valid signature($\chi' = K_2 * \psi' + K_3$), then **R** outputs $(\psi' - K_1)$; else (that is if either the signature is invalid($\chi' \neq K_2 * \psi' + K_3$) or the original message is not received), **R** *knows* the identity (among the two possibilities of α or β) of the set that is the corrupt set. How? This is because, the path Q completely avoids the nodes from one of these sets say B_j, $j \in \{\alpha, \beta\}$; this clearly means that a faulty path Q (since a wrong message was delivered) entails that set $B_{\bar{j}}$ is corrupt (where $\bar{j} = \{\alpha, \beta\} - \{j\}$).

Thus, what the above sub-protocol achieves is the following (under the big assumption that the adversary can corrupt one among only the two sets B_α or B_β): If the set B_j, $j \in \{\alpha, \beta\}$, is not corrupt (which means that the other set may be corrupt), then **R** receives the correct message with certainty while the adversary has no information about the message. On the other hand, if the set B_j is corrupt, then though the adversary still has no information about the transmitted message, he has complete control over **R**'s output. **R**'s output could therefore either be a valid message or a null message with the knowledge that (any subset of) B_j is corrupt. Moreover, if **R** receives a valid message, it is the correct message with a very high probability.

Now, since there are $(t+1)$ strong paths from **S** to **R**, one of them must avoid B_j. Thus if **S** sends the message along all these paths, the knowledge that B_j is corrupt is sufficient for **R** to recover the correct message. Thus, if **R** must not receive the message yet, he must not know the identity of the corrupted set which in turn means that we have, with a very high probability, simulated an edge from w_1 to **R** (this simulation fails only when **R** is able to get **S**'s message which is what we are anyway striving for). Assuming this edge from w_1 to **R**, we find that the number of nodes along the path p such that it has both the adjacent edges directed in-ward towards itself has effectively reduced by one and induction follows.

[1] Since the adversary may corrupt only upto t nodes, at least one of $t+1$ paths must be honest.

This completes our exercise of constructing the sub-protocol $\Pi_{\alpha\beta}$ that is guaranteed to work correctly only if one of B_α or B_β is chosen by the adversary.

Using all the sub-protocols, $\Pi_{\alpha\beta}$'s, $1 \leq \alpha, \beta \leq \binom{n}{t}$ and $\alpha \neq \beta$, to design the actual protocol Π: It is clear from the prequel that if the network is $(2t,t)$-(**S**,**R**)-special-connected as well as $(t+1)$-(**S**,**R**)-strong-connected, then *all* the $\binom{\binom{n}{t}}{2}$ $\Pi_{\alpha\beta}$ sub-protocols will exist. We now construct the final $(t,(1-\delta))$-reliable URMT protocol Π using these sub-protocols. The construction is as follows: **S** and **R** execute all the $\binom{\binom{n}{t}}{2}$ $\Pi_{\alpha\beta}$ sub-protocols with the same message M. At the end of all these sub-protocols, **R** would have received $\binom{\binom{n}{t}}{2}$ values. We now show how **R** can recover M locally using these values.

Note that **R** can simulate the sub-protocol $\Pi_{\alpha\beta\gamma}$ which assumes that one among the *three* sets B_α or B_β or B_γ is chosen by the adversary. The simulation is done as follows: **R** takes the majority among the outputs of the three protocols $\Pi_{\alpha\beta}$, $\Pi_{\beta\gamma}$ and $\Pi_{\alpha\gamma}$. A majority is bound to exist since any set chosen by the adversary is tolerated in *two* of the three protocols. Next, R can simulate the sub-protocol which behaves like a URMT protocol as long as any one among a collection of *four* sets is chosen by the adversary. Continuing further, **R** will be able to simulate the protocol that behaves correctly if one among the collection of $\binom{n}{t}$ sets is chosen by the adversary. This protocol by definition is a $(t,(1-\delta))$-reliable URMT protocols from **S** to **R**!

This completes the sufficiency part of the proof of the theorem.

Necessity: The necessity of $(t+1)$-(**S**,**R**)-strong-connectivity is obvious for otherwise the adversary can disconnect **R** from **S**. What is left is to show the necessity of $(2t,t)$-(**S**,**R**)-special-connectivity.

We show that if the network is not $(2t,t)$-(**S**,**R**)-special-connected then no URMT protocol from **S** to **R** can exist. By definition, in a non-$(2t,t)$-(**S**,**R**)-special-connected network, one can find a two sets D_1 and D_2, $|D_1| \leq t$ and $|D_2| \leq t$ such that the deletion of $(D_1 \cup D_2)$ has one of the following two effects:

- it removes all weak paths from **S** to **R**
- every remnant weak path from **S** to **R** has a node w such that w has both its adjacent edges directed inwards and both D_1 and D_2 cut across all strong paths from w to **R**.

In either of the above cases, we show the impossibility of URMT from **S** to **R**. Define the set $Y \subset \mathbb{P}$ as the set of vertices that have a strong path to **R** in $\mathcal{N}$ that does not use any vertex from $(D_1 \cup D_2)$. Furthermore, let $X = \mathbb{P} \setminus (D_1 \cup D_2 \cup Y)$. Since the sender **S** and the receiver **R** are honest, clearly $\mathbf{S} \in X$ and $\mathbf{R} \in Y$. Moreover, it is evident from the definition of Y that there do not exist vertices $u \in X$ and $v \in Y$ such that the edge (u,v) is in $\mathcal{N}$. We first consider the first of the aforementioned effects, namely, that there do not exist vertices $u \in X$ and $v \in Y$ such that the edge (v,u) is in $\mathcal{N}$.

LEMMA 4.1. *The conditions of the Theorem 4.1 are necessary for the existence of URMT over network $\mathcal{N}$ if there do not exist vertices $u \in X$ and $v \in Y$ such that the edge (v,u) is in $\mathcal{N}$.*

Proof. We will prove the impossibility even for the best case where every other edge (other than those between X and Y) exists. Define two executions $\mathbf{E}_0$ and $\mathbf{E}_1$ as follows. In both executions the vertices in Y hold the random inputs $\{\rho_u | u \in Y\}$. In the execution $\mathbf{E}_\alpha \in \{\mathbf{E}_0, \mathbf{E}_1\}$, the Byzantine set D_α is corrupt and the message m_α is transmitted by **S**, the random inputs of the vertices in $(X \cup D_{\overline{\alpha}})$[2] are $\{\rho_u | u \in (X \cup D_{\overline{\alpha}})\}$. The behavior of the Byzantine set D_α in the execution $\mathbf{E}_\alpha$ is to send no message whatsoever to $X \cup D_{\overline{\alpha}}$ and to send to Y exactly the same messages that are sent to Y by the honest D_α in the execution $\mathbf{E}_{\overline{\alpha}}$. In order for the Byzantine set D_α to behave as specified in the execution $\mathbf{E}_\alpha$, the adversary needs to simulate the behavior of $(X \cup D_\alpha)$ in the execution $\mathbf{E}_{\overline{\alpha}}$. To achieve this task, the adversary simulates round-by-round the behavior of the vertices in $(X \cup D_\alpha)$ for the execution $\mathbf{E}_{\overline{\alpha}}$ using $\{\rho_u | u \in (X \cup D_\alpha)\}$ as the random inputs for the vertices in $(X \cup D_\alpha)$. At the beginning of each round, each simulated player has a history of messages that it got in the simulation of the previous rounds and its simulated local random input. The simulated player sends during the simulation the same messages that the honest player would send in the original protocol in the same state. The simulated messages that (players in) D_α sends to **R** are really sent by the players. All other messages are used only to update the history for the next round. The messages which are added to the history of each simulated vertex are the real messages that are sent by players in Y and the simulated messages that are sent by the vertices in $(X \cup D_\alpha)$. No messages from $D_{\overline{\alpha}}$ are added to history. The history of messages of each simulated vertex in execution $\mathbf{E}_\alpha$ is the same as the history of the vertex in execution $\mathbf{E}_{\overline{\alpha}}$. Therefore, the messages sent by D_1 and D_2 to members of Y in both

[2] We denote $\overline{1} = 2$ and viceversa.

executions are exactly the same and the members of Y and in particular the receiver $\mathbf{R}$ receive and send the same messages in both executions. Thus, the receiver $\mathbf{R}$ cannot distinguish whether the set D_1 is corrupt and the message transmitted by $\mathbf{S}$ is m_1 or the set D_2 is corrupt and the message transmitted by $\mathbf{S}$ is m_2. Now, consider all the pairs of executions where the random inputs range over all possible values. In each pair of executions, whenever $\mathbf{R}$ accepts the correct message in one execution it commits an error in the other. Thus, for any strategy by $\mathbf{R}$ for choosing whether to receive m_1 or m_2 there is some α such that when m_α is transmitted, the receiver accepts m_α with probability at most $\frac{1}{2}$. This completes the proof of Lemma 4.1. □

Now, we turn our attention to the case of the second effect, namely if a weak path exists but has a node w with both its adjacent edges directed inward as well as has both D_1 and D_2 disconnecting $\mathbf{R}$ from w. We will now prove that every such weak path between X and Y is essentially "useless" thereby maintaining the impossibility of URMT as projected by the Lemma 4.1. At least one edge from these weak paths must be from a node in Y to another node in X (since these are paths outside $(D_1 \cup D_2)$ and from $\mathbf{S}$ to $\mathbf{R}$). We will show that removing that edge has no effect of the possibility of URMT thereby proving the required result.

Firstly, how can these edges be useful? The answer is that they can be used by players in Y to send some secret messages to the players in X such that the adversary, oblivious of these messages, cannot simulate the messages of X without being distinguished by Y. However, if we are able to show that no such secret information can help URMT from $\mathbf{S}$ to $\mathbf{R}$, then we are through. We do the same now.

A node x is said to have no influence on $\mathbf{R}$ if the output of $\mathbf{R}$ is independent of values sent by x. Otherwise x is said to influence $\mathbf{R}$. Consider an edge (y, x) in $\mathcal{N}$ such that $y \in Y$ and $x \in X$. We need to know whether x can influence $\mathbf{R}$ by using the data received from y. Suppose we manage to show that it cannot then we are through since what it means is that data sent along the edge (y, x) has no effect on $\mathbf{R}$ and hence can be ignored. We now proceed to prove the same.

Suppose that the node $\mathbf{R}$ can be influenced by x. This (at least) means that there must be a path $x, w_1, w_2, \ldots, w_q, \mathbf{R}$ in $\mathcal{N}$ such that x transmits some information to w_1, then w_1 transmits some information to w_2 that depends on the information it got from x and so on until some information gets to $\mathbf{R}$.[3]

LEMMA 4.2. *If there exists an $\alpha \in \{1, 2\}$ such that every strong path from x to $\mathbf{R}$ in $\mathcal{N}$ passes through some node(s) in (D_α)* followed *by some node(s) in $D_{\overline{\alpha}}$, then x cannot influence $\mathbf{R}$ (using the data received from y).*

Proof. Given that every path from x to $\mathbf{R}$ passes through some node(s) in D_α *followed* by some node(s) in $D_{\overline{\alpha}}$ for some $\alpha \in \{1, 2\}$, if the adversary corrupts the α^{th} set in $\mathcal{A} = \{D_1, D_2\}$, does the following: let w_j be the first vertex in D_α on a path from x to $\mathbf{R}$. The corrupt w_j ignores the real messages that it gets from the players in $X \cup D_{\overline{\alpha}}$ and thus the messages that it sends do not depend on the messages sent by x. Similarly, the messages sent by x when D_α simulates the players in X do not influence the messages it sends to $\mathbf{R}$ since the path from x to $\mathbf{R}$ passes through at least one vertex from $D_{\overline{\alpha}}$ and no messages are sent by players in $D_{\overline{\alpha}}$ during the simulation. Thus even if $\mathbf{R}$ may know that the correct secret (that was exchanged using the edge (y, x)) was not used, he will not know which set in $\mathcal{A}$ to blame. Hence the lemma. □

Note that we actually *know* that every path from $x \in X$ to $\mathbf{R}$ passes through some node(s) in (D_α) *followed* by some node(s) in $D_{\overline{\alpha}}$ for some $\alpha \in \{1, 2\}$. Otherwise, it contradicts the definition of $(2t, t)$-special-connectivity. Thus the simulated messages of x have no influence on the messages received by $\mathbf{R}$ and can be ignored. Hence, the impossibility of URMT proved in Lemma 4.1 is not altered by using the edges from Y to X. □

5 URMT versus PRMT

THEOREM 5.1. *For any positive integer $k > 0$, there always exists a digraph G_k such that $(k, 1)$-reliable URMT is* impossible *over G_k whereas there exists a $(2k, (1 - \delta))$-reliable URMT protocol, $\delta > 0$ in G_k.*

Proof. For any given $k > 0$, we construct a graph $G_k = (V, E_1 \cup E_2)$ with

$$V = \{v_1, v_2, v_3, \ldots, v_{2k+1}, \mathbf{S}, \mathbf{R}\}$$

$$E_1 = \{(\mathbf{S}, v_1), (v_1, \mathbf{R}), \ldots (\mathbf{S}, v_{k+1}), (v_{k+1}, \mathbf{R})\}$$

$$E_2 = \{(v_{k+2}, \mathbf{S}), (v_{k+2}, \mathbf{R}), \ldots (v_{2k+1}, \mathbf{S}), (v_{2k+1}, \mathbf{R})\}$$

[3] Since the network is synchronous, it may be possible to transmit information without actually sending message bits. However, even such transmissions are possible only between nodes that can actually exchange some message-bits as well. Thus, an information-path is necessarily a physical path too.

The graph for $k = 2$ is illustrated in figure 2 for clarity.

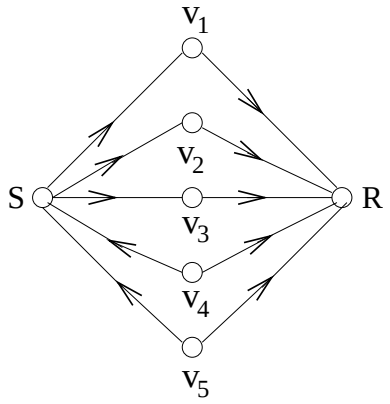

Figure 2: The Graph G_2 for $k = 2$.

The graph G_k thus constructed is clearly $(k+1)$-(**S,R**) strong connected. Hence $(\frac{k}{2}, 1)$-reliable URMT is possible, while $(k, 1)$-reliable URMT[4] is impossible over G_k. But it is $(2k, k)$-(**S,R**)-special-connected, hence by our characterization in section 4, $(k, (1-\delta))$ -reliable URMT protocol is possible over G_k. □

THEOREM 5.2. *The factor* 2 *in the theorem 5.1 is tight.*

Proof. Suppose there exists a $((2+\epsilon)k, (1-\delta))$ -reliable URMT protocol on any G_k, $\epsilon > 0$, it implies that G_k should be at least $(2+\epsilon)k$-(**S,R**)-strong-connected. Thus, $(k, 1)$-reliable URMT will be possible over G_k. □

Consider the network shown in the Figure 2. Note that the graph is 3-(**S,R**)-strong-connected and (4-2)(**S,R**)-special-connected. Therefore, the given network can tolerate adversary of size one in case of PRMT whereas it can tolerate adversary of size 2 in the case of URMT. Thus, it is clear that the gap is significantly high in terms of tolerability of adversary between PRMT and URMT protocols.

6 Digraphs with $\Omega(n^2)$ Critical Edges

DEFINITION 6.1. *In a digraph G on which $(t, (1-\delta))$-reliable URMT is possible, an edge is called* critical *if the deletion of that edge renders $(t, (1-\delta))$-reliable URMT impossible.*

In the case of PRMT, network is abstracted to $(2t+1)$ wires and the message is transmitted along those wires where t is the maximum number of nodes that can be corrupted by an adversary. Therefore, the critical edges in case of PRMT are those edges that are part of the abstracted $(2t+1)$ wires. Thus, the number of critical edges is always $O(n)$ edges.

Similarly in the case of URMT in undirected graphs, from the results of [3], we know that the network is again abstracted to $(2t+1)$ wires. Following similar arguments as in the previous case of PRMT, we can show that even in the case of URMT in undirected graphs, any graph can have only $O(n)$ critical edges. However, surprisingly, we show that in the case of URMT in directed graphs, for *every* n there exist networks even with $\Omega(n^2)$ critical edges.

For any given $n > 3$, we construct a graph $G_t = (V, E_1 \cup E_2 \cup E_r)$ with

$$V = \{v_1, v_2, v_3, \ldots, v_{2t+1}, \mathbf{S}, \mathbf{R}\}$$

$$E_1 = \{(\mathbf{S}, v_1), (v_1, \mathbf{R}), \ldots, (\mathbf{S}, v_{t+1}), (v_{t+1}, \mathbf{R})\}$$

$$E_2 = \{(\mathbf{S}, v_{t+2}), (\mathbf{R}, v_{t+2}), \ldots (\mathbf{S}, v_{2t+1}), (\mathbf{R}, v_{2t+1})\}$$

$$E_r = \bigcup_{i=0}^{t-1} \{(v_{t+2+i}, v_1), (v_{t+2+i}, v_2), \ldots, (v_{t+2+i}, v_{t+1})\}$$

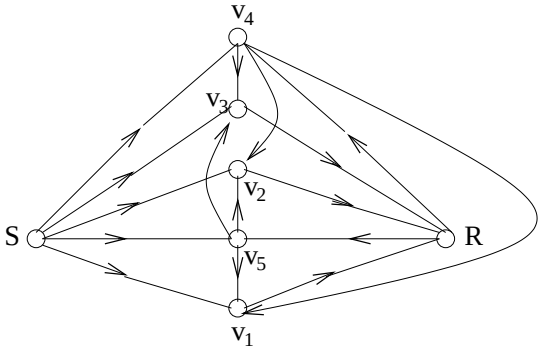

Figure 3: The Graph G_2 for $t = 2$.

Consider the example shown in figure 3. G_2 is 3-(**S,R**) strong-connected and (4,2)-(**S,R**) special-connected. However, observe that each edge is critical as removal of even one edge makes G_2 intolerable to any $(2, (1-\delta))$-reliable URMT protocol. Based on similar arguments, we may see that the total number of edges required in the graph G_t is $t(t+1)+2(2t+1)$. Since t is $\lfloor \frac{n-3}{2} \rfloor$ in G_t, the number of *critical edges* in G_t is $\Omega(n^2)$.

[4]The graph G_k is not $(2k+1)$-(**S,R**) strong connected.

7 Efficient Protocols for All-Pairs URMT

We show how to design *efficient* $(t, (1-\delta))$-reliable URMT protocols from $\mathbf{S}$ to $\mathbf{R}$ for the class of digraphs that support a $(t, (1-\delta))$-reliable URMT protocol between any two nodes in the digraph/network. First, it is obvious that such a connected network must have at least $(2t+1)$ vertex disjoint weak paths from $\mathbf{S}$ to $\mathbf{R}$ and also $(t+1)$ vertex disjoint strong paths from $\mathbf{S}$ to $\mathbf{R}$ (note that these two sets of vertex disjoint paths may have several common vertices between them). Let the vertex disjoint weak paths be denoted by $q_1, q_2, \ldots, q_{2t+1}$ (ignoring the others if there are more that $(2t+1)$ such paths) and the vertex disjoint strong paths be denoted by $p_1, p_2, \ldots, p_{t+1}$ (note that a path from the p_i's and another from the q_i's may intersect each arbitrarily without any restriction).

Next, we show how to design an efficient $(t, (1-\delta))$-reliable URMT protocol Π_i assuming that there exists an i such that no nodes occurring in the path q_i is corrupted. In other words, Π_i would work for us if we were lucky enough that the adversary chose to corrupt all the t nodes outside of the nodes in the path q_i.

Subsequently, we will use the $(2t+1)$ sub-protocols, namely Π_i's, $1 \leq i \leq 2t+1$, to construct a new protocol Π that would then be proved to be a $(t, (1-\delta))$-reliable URMT protocol.

Designing the efficient sub-protocol Π_i*:* If the path p_i does not contain any nodes with both its adjacent edges directed inward, then the protocol designed in the Case 1 of the sufficiency proof of Theorem 4.1 works for us. However, if there do exist such a node, say w, in the path p_i, in a general graph we were unable to help much but here we know that w has $t+1$ vertex disjoint strong paths to $\mathbf{R}$. Therefore, the protocol in Case 2 of the sufficiency proof works for *all* the B_α and B_β pairs that avoid nodes from the path p_i. Thus, instead of $\binom{\binom{n}{t}}{2}$ different protocols, we can cover the same with $O(|\mathbb{P}|)$ distinct protocols.

Using Π_i*'s to design the efficient protocol* Π*:* In this case, there are only a linear number of efficient protocols to be executed which can help $\mathbf{R}$ recover the correct message M in an analogous manner as was done when super-polynomial sub-protocols were executed.

Note that using the above technique it is possible to design efficient URMT protocols in all cases wherein there are $t+1$ vertex disjoint paths from w to $\mathbf{R}$, were w represents a node in a weak path with both its adjacent edges directed inward.

8 Concluding Remarks

The characterizations of the possibility of reliable message transmission over synchronous networks in the extant literature include the well-known $(2t+1)$-connectivity for PRMT in undirected graphs [2], $(2t+1)$-connectivity again for URMT in undirected graphs [3] and $(2t+1)$-strong-connectivity for PRMT in directed graphs [1]. Note that all of the above are quite pleasing to the mind and of course, more importantly, easy to verify on any give input. However, the truth for URMT in directed graphs, for the first time in the history of reliable communication problem, is marked different from the above simple characterizations as we have shown in this work. We leave it as an interesting open problem to design efficient algorithms to compute the maximum t such that a given digraph is $(2t, t)$-($\mathbf{S}$,$\mathbf{R}$)-special-connected. At first glance, this does not appear to be easy and perhaps it may even be NP-Hard. The fundamental and inherent complexity of $(2t, t)$-($\mathbf{S}$,$\mathbf{R}$)-special-connectivity not only makes it difficult to verify it but also affects the design of general purpose efficient URMT protocols. Again, since one cannot yet rule-out the possibility of a super-polynomial lower bound on the complexity of general URMT protocols, a study in that direction may be worthwhile. Yet another interesting line of research is from an extremal graph theoretic viewpoint wherein we may ask "for a given t, what is the minimum number of edges required by an n node digraph such that URMT is possible over it". Finally, it is also interesting to study the relative difficulty of computing the optimal round complexity of URMT among various classes of graphs.

References

[1] 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.

[2] 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.

[3] M. Franklin and R.N. Wright. Secure Communication in Minimal Connectivity Models. *Journal of Cryptology*, 13(1):9–30, 2000.

[4] 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.