Asynchronous Throughput-Optimal Routing in Malicious Networks[*]

Paul Bunn[1,**] and Rafail Ostrovsky[1,2,***]

[1] UCLA Department of Mathematics
paulbunn@math.ucla.edu
[2] UCLA Department of Computer Science
rafail@cs.ucla.edu

Abstract. We demonstrate the feasibility of throughput-efficient routing in a highly unreliable network. Modeling a network as a graph with vertices representing nodes and edges representing the links between them, we consider two forms of unreliability: unpredictable edge-failures, and deliberate deviation from protocol specifications by corrupt nodes. The first form of unpredictability represents networks with dynamic topology, whose links may be constantly going up and down; while the second form represents malicious insiders attempting to disrupt communication by deliberately disobeying routing rules in an arbitrary manner, for example by introducing junk messages or deleting or altering messages. We present a robust routing protocol for end-to-end communication that is simultaneously resilient to both forms of unreliability, achieving provably optimal throughput performance.

Keywords: Network Routing, Fault Localization, Multi-Party Computation in Presence of Dishonest Majority, Communication Complexity, End-to-End Communication, Competitive Analysis, Asynchronous Protocols.

1 Introduction

With the immense range of applications and the multitude of networks encountered in practice, there has been an enormous effort to study routing in various settings. For the purpose of developing network models in which routing protocols can be developed and formally analyzed, networks are typically modeled as a graph with vertices representing nodes (processors, routers, etc.) and edges representing the connections between them. Beyond this basic structure, additional assumptions and restrictions are then made in attempt to capture various

[*] Full version of the paper is available on-line [18].
[**] Supported in part by Lockheed Martin Corporation and NSF grants 0716835, 0716389, 0830803, 0916574.
[***] Supported in part by Lockheed Martin Corporation, IBM Faculty Award, Xerox Innovation Group Award, the Okawa Foundation Award, Intel, Teradata, NSF grants 0716835, 0716389, 0830803, 0916574 and U.C. MICRO grant.

features that real-world networks may display. In deciding which network model is best-suited to a particular application, developers must make a choice with respect to each of the following considerations: 1) Synchronous or Asynchronous; 2) Static or Dynamic Topology; 3) Global Control or Distributed/Local Control; 4) Connectivity/Liveness Assumptions; 5) Existence of Faulty/Malicious Nodes.

Notice that in each option above there is an inherent trade-off between generality/applicability of the model verses optimal performance within the model. For instance, a protocol that assumes a fixed network topology will likely out-perform a protocol designed for a dynamic topology setting, but the former protocol may not work in networks subject to edge-failures. Similarly, a protocol that protects against the existence of faulty or deliberately malicious nodes will likely be out-performed in networks with no faulty behavior by a protocol that assumes all nodes act honestly.

From both a theoretical and a practical standpoint, it is important to understand how each (combination) of the above listed factors affects routing performance. In this paper, we explore the feasibility of end-to-end routing in highly unreliable networks, i.e. networks that simultaneously consider *all* of the more general features: Asynchronous, Dynamic Topology, Local Control, no Connectivity Assumptions, and the presence of Malicious Nodes. In this "worst-case" model it is unlikely that any protocol will perform well, and stronger assumption(s) must be made to achieve a reasonable level of performance. However, understanding behavior in the worst case, even with respect to the most basic task of end-to-end communication, is important to determine how much (if any) the addition of each assumption improves optimal protocol performance.

1.1 Previous Work

As mentioned above, development and analysis of routing protocols relies heavily on the choices made for the network model. To date, all network models have guaranteed at least one (and more commonly multiple) "reliability" assumption(s) with respect to the above list of five network characteristics. In this section, we explore various combinations of assumptions that have been made in recent work, highlighting positive and negative results with respect to each network model, and emphasizing clearly which assumptions are employed in each case. Since our work focuses on theoretical results, for space considerations we do not discuss below the vast amount of research and analysis of routing issues for specific network systems encountered in practice, e.g. the Internet. Still, the amount of research regarding network routing and analysis of routing protocols is extensive, and as such we include only a sketch of the most related work, indicating how their models differ from ours and providing references that offer more detailed descriptions.

End-to-End Communication: One of the most relevant research directions to our paper is the notion of End-to-End communication in distributed networks, where two nodes (sender and receiver) wish to communicate through a network. While there is a multitude of problems that involve end-to-end communication (e.g. End-to-End Congestion Control, Path-Measurement, and Admission Control), we discuss here work that consider networks whose only task is to facilitate communication between sender and receiver. Some of these include a line of work developing the *Slide* protocol (the starting point of our protocol) of Afek, Gafni and Rosén [3] (see also Afek et al. [1].) The Slide protocol (and its variants) have been studied in a variety of network settings, including multi-commodity flow (Awerbuch and Leighton [11]), networks controlled by an online bursty adversary (Aiello et al. [4]), and networks that allow corruption of nodes (Amir et al. [7]). However, prior to our work there was no version of Slide that considered routing in the "worst case" network setting: only [7] considers networks with corruptible nodes, but their network model assumes synchronous communication and demands minimal connectivity guarantees.

Fault Detection and Localization Protocols: There have been a number of papers that explore the possibility of corrupt nodes that deliberately disobey protocol specifications in order to disrupt communication. In particular, there is a recent line of work that considers a network consisting of a *single path* from the sender to the receiver, culminating in the recent work of Barak et al. [13] (for further background on fault localization see references therein). In this model, the adversary can corrupt any node (except the sender and receiver) in a dynamic and malicious manner. Since corrupting any node on the path will sever the honest connection between sender and receiver, the goal of a protocol in this model is *not* to guarantee that all messages sent are received. Instead, the goal is to *detect* faults when they occur and *localize* the fault to a single edge.

Goldberg et al. [19] show that a protocol's ability to detect faults relies on the assumption that One-Way Functions (OWF) exist, and Barak et al. [13] show that the (constant factor) overhead (in terms of communication cost) incurred for utilizing cryptographic tools (such as MACs or Signature Schemes) is mandatory for any fault-localization protocol. Awerbuch et al. [10] also explore routing in the Byzantine setting, although they do not present a formal treatment of security, and indeed a counter-example that challenges their protocol's security is discussed in the appendix of [13].

Fault Detection and Localization protocols focus on very restrictive network models (typically synchronous networks with fixed topology and some connectivity assumptions), and throughput-performance is usually not considered when analyzing fault detection/localization protocols.

Competitive Analysis: Competitive Analysis was introduced by Sleator and Tarjan [22] as a mechanism for measuring the *worst-case* performance of a protocol, in terms of how badly the given protocol may be out-performed by an *off-line* protocol that has access to perfect information. Recall that a given protocol has *competitive ratio* $1/\lambda$ (or is λ-*competitive*) if an ideal off-line protocol has advantage over the given protocol by at most a factor of λ. One place competitive analysis has been used to evaluate performance is the setting of distributed algorithms in asynchronous shared memory computation, including the work of Ajtai et al. [6]. This line of work has a different flavor than the problem considered in the present paper due to the nature of the algorithm being analyzed (computation algorithm verses network routing protocol). In particular, network topology is not a consideration in this line of work (and malicious deviation of processors is not considered). Competitive Analysis also plays a role in Adversarial Queuing Theory, see, for example [15]. (We discuss this aspect in greater detail below.)

Competitive analysis is a useful tool for evaluating protocols in unreliable networks (e.g. asynchronous networks and/or networks with no connectivity guarantees), as it provides best-possible standards (since absolute performance guarantees may be impossible due to the lack of network assumptions). For a thorough description of competitive analysis, see [14].

Max-Flow and Multi-Commodity Flow: The Max-flow and multi-commodity flow models assume synchronous networks with connectivity/liveness guarantees and have incorruptible nodes (max-flow networks also typically have fixed topology and are global-control). There has been a tremendous amount of work in these areas, see e.g. Leighton et al. [21] for a discussion of the two models and a list of results, as well as Awerbuch and Leighton [11] who show optimal throughput-competitive ratio for the network model in question.

Admission Control and Route Selection: The admission control/route selection model differs from the multi-commodity flow model in that the goal of a protocol is not to meet the demand of all ordered pairs of nodes (s, t), but rather the protocol must decide which requests it can/should honor, and then designate a path for honored requests. There are numerous models that are concerned with questions of admission control and route selection: The Asynchronous Transfer Model (see e.g. Awerbuch et al. [9]), Queuing Theory (see e.g. Borodin and Kleinberg [15] and Andrews et al. [8]), Adversarial Queuing Theory (see e.g. Broder et al. [16] and Aiello et al. [5]).

The admission control/route selection model assumes synchronous communication and incorruptible nodes and makes connectivity/liveness guarantees. Among the other options (fixed or dynamic topology, global or local control), each combination has been considered by various authors, see the above references for further details and results within each specific model.

1.2 Our Results

In this paper, we consider the feasibility of end-to-end routing in unreliable networks controlled by a malicious adversary with polynomial computing power. In particular, we present a local-control protocol that achieves optimal throughput in asynchronous networks with untrustworthy nodes and dynamic topology with no connectivity guarantees.

The first step in obtaining our result is to first consider networks where nodes are guaranteed to behave honestly, but otherwise the network demonstrates all of the unreliability features. In these networks, we have the following matching upper and lower bounds on throughput performance:

Theorem 1 (Informal). *The best competitive-ratio that any protocol can achieve in a distributed asynchronous network with dynamic topology (and no connectivity assumptions) is* $1/n$ *(where* n *is the number of nodes). In particular, given any protocol* $\mathcal{P}$*, there exists an alternative protocol* $\mathcal{P}'$*, such that* $\mathcal{P}'$ *will out-perform* $\mathcal{P}$ *by a factor of at least* n*.*

Theorem 2 (Informal). *There exists a protocol that achieves a competitive ratio of* $1/n$ *in a distributed asynchronous network with dynamic topology (and no connectivity assumptions).*

Due to space constraints, we do not prove Theorems 1 and 2 here, but refer the reader to [17] for full details of the proofs of each of these. In this paper, we focus on the following result, which extends Theorem 2 to networks in which nodes are no longer assumed to behave honestly; i.e. they may deviate from the specified protocol in any desired manner to disrupt communication as much as possible. Somewhat surprisingly, we show that this increased level of unreliability does not affect optimal throughput performance; indeed, we demonstrate a protocol that achieves $1/n$ competitive ratio, matching the lower-bound of Theorem 1.

Theorem 3 (MAIN THEOREM, Informal). *Assuming Public-Key Infrastructure, there exists a protocol with competitive ratio* $1/n$ *in a distributed asynchronous network with dynamic topology (and no connectivity assumptions), even if a polynomially bounded adversary can dynamically corrupt an arbitrary set of nodes.*

2 The Model

In this section, we describe formally the model in which we will be analyzing routing protocols. We begin by modeling the network as a graph G with n vertices (or *nodes*). Two of these nodes are designated as the *sender* and *receiver*, and the sender has a stream of messages $\{m_1, m_2, \dots\}$ that it wishes to transmit through the network to the receiver.

Asynchronous communication networks vary from synchronous networks in that the transmission time across an edge in the network is not fixed (even along the same edge, from one message transmission to the next). Since there is no common global clock or mechanism to synchronize events, an asynchronous network is often said to be "message driven," in that the actions of the nodes occur exactly (and only) when they have just sent/received a message.

Asynchronous networks are commonly modeled by introducing a scheduling adversary that controls the edges of the network as follows. Informally, we focus on a single edge $E(u, v)$, and then a "round" consists of allowing the edge to deliver a message in both directions.[1] To model unpredictable delivery times across each edge, we have each node u decide on the next message to send to v

[1] An adversary that is allowed to deliver messages in only *one* direction can be modelled by defining a round to consist of (at least) one communication in each direction. Since competitive analysis can be used to show that acknowledgements of some kind are requisite for a finite competitive-ratio, it is natural to define a round as above.

immediately after receiving a message from v, and this message is then sent to the adversary who stores it until the next time he activates edge $E(u, v)$.

Formally, we define a round to consist of a single edge $E(u, v)$ in the network chosen by the adversary in which two sequential events occur: 1a) Among the packets from u to v that the adversary is storing, it will choose one (in any manner it likes) and deliver it to v; 1b) Similarly, the adversary chooses one of the packets it is storing from v to u and delivers it to u; 2a) After seeing the delivered packet, u sends requests of the form $(u, v, m) =$ (sending node, target node, message) to the adversary, which will be stored by the adversary and may be delivered the *next* time $E(u, v)$ is made a round; 2b) Similarly for v.

Modeling asynchronicity in this manner captures the intuition that a node has no idea how long a message "sent" to an adjacent node will take to arrive, and this definition also captures the "worst-case" asynchronicity, in that a (potentially deliberately malicious) adversary controls the scheduling of rounds/edges.

Aside from obeying the above specified rules, we place no restriction on the scheduling adversary. In other words, it may activate whatever edges it likes (this models the fact our network makes no connectivity assumptions), wait indefinitely long between activating the same edge twice (modeling both the dynamic and asynchronous features of our network), and do anything else it likes (so long as it respects steps (1) and (2) above each time it activates an edge) in attempt to hinder the performance of a routing protocol.

Our model also allows a polynomially bounded node-controlling adversary to corrupt the *nodes* in the network. The node-controlling adversary is malicious, meaning that takes complete control over the nodes he corrupts, and can therefore force them to deviate from any protocol in whatever manner he likes. We also allow for a dynamic adversary, which means that he can corrupt nodes at any stage of the protocol, deciding which nodes to corrupt based on what he has observed thus far. We do not impose any "access-structure" limitations on the adversary. That is, the adversary may corrupt any nodes it likes (although if the sender and/or receiver is corrupt, secure routing between them is impossible). Because integrity of the messages received by the receiver is now a concern (as corrupt nodes can delete and/or modify messages), we will say a routing protocol is secure if the receiver eventually gets all of the messages sent by the sender, in order and without duplication or modification.

The separation of the adversaries into two distinct entities is solely for conceptual reasons. Note that the scheduling adversary cannot be controlled or eliminated: edges themselves are not inherently "good" or "bad," so identifying an unresponsive edge does not allow us to forever refuse the protocol to utilize this edge. By contrast, our protocol will limit the amount of influence the node-controlling adversary has in the network. Specifically, we will show that if a node deviates from the protocol in a sufficiently destructive manner (in a well-defined sense), then our protocol will be able to identify it as corrupted in a timely fashion. Once a corrupt node has been identified, it will be eliminated from the network by excluding it from all future communication.

Note that our network model is on-line and distributed, in that we do not assume that the nodes have access to any information (including future knowledge of the adversary's schedule) aside from the packets they receive. Also, we insist that nodes have bounded memory which is at least $\Omega(n^2)$.[2]

Our mechanism for evaluating protocols will be to measure their throughput, a notion we can now define formally in the context of rounds and the scheduling adversary. In particular, let $f_{\mathcal{P}}^{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$ be a function that measures, for a given protocol $\mathcal{P}$ and adversary $\mathcal{A}$, the number of packets that the receiver has received as a function of the number of rounds that have passed. Note that in this paper, we will consider only deterministic protocols, so $f_{\mathcal{P}}^{\mathcal{A}}$ is well-defined. The function $f_{\mathcal{P}}^{\mathcal{A}}$ formalizes our notion of throughput.

As mentioned in the Introduction, we utilize competitive analysis to gauge the performance (with respect to throughput) of a given protocol against all possible competing protocols. In particular, for any fixed adversary $\mathcal{A}$, we may consider the ideal "off-line" protocol $\mathcal{P}'$ which has perfect information: knowledge of all future decisions of the scheduling adversary, as well as knowledge of which nodes are/will become corrupt. That is, for any fixed round x, there exists an ideal off-line protocol $\mathcal{P}'(\mathcal{A}, x)$ such that $f_{\mathcal{P}'}^{\mathcal{A}}(x)$ is maximal. We demand that the ideal protocol $\mathcal{P}'$ never utilizes corrupt nodes once they have been corrupted (this restriction is not only reasonable, it is necessary, as it can easily be shown that allowing $\mathcal{P}'$ to utilize corrupt nodes will result in *every* on-line protocol having competitive ratio $1/\infty$).

Definition 1. *We say that a protocol $\mathcal{P}$ has competitive ratio $1/\lambda$ (respectively is λ-competitive) if there exists a constant k and function $g(n, C)$ (C is the memory bound per node) such that for all adversaries $\mathcal{A}$ and for all $x \in \mathbb{N}$:*[3]

$$f_{\mathcal{P}'}^{\mathcal{A}}(x) \ \leq\ (k \cdot \lambda)\ \cdot f_{\mathcal{P}}^{\mathcal{A}}(x)\ +\ g(n, C) \tag{1}$$

We assume a Public-Key Infrastructure (PKI) that allows digital signatures. In particular, this allows the sender and receiver to sign messages to each other that cannot be forged (except with negligible probability in the security parameter) by any other node in the system. It also allows nodes to verify/sign communications with their neighbors (see Section 3.2).

3 Description of Protocol

In this section we present an on-line protocol that enjoys competitive ratio $1/n$ in the network model of Section 2. Our protocol uses as its starting point the "Slide" protocol (or *gravitational-flow*), which was first introduced by Afek, Gafni, and Rosén [3], and further developed in a series of work [1], [20], and [17]. As is shown in [17], Slide+ enjoys competitive ratio $1/n$ in networks in which all nodes behave

[2] For simplicity, we assume that all nodes have the same memory bound, although our argument can be readily extended to handle the more general case.

[3] Typically, λ is a function of the number of nodes in the network n, and Definition 1 implicitly assumes the minimal value of λ for which (1) holds.

honestly (but otherwise the network is as modelled here). We first describe this protocol in Section 3.1, and then in Section 3.2 describe how we modify this protocol to address networks allowing misbehaving nodes.

3.1 Description of the Slide+ Protocol

Recall that we model an asynchronous network via a scheduling adversary that maintains a buffer of requests of the form (u, v, p), which is a request from node u to send packet p to node v. The scheduling adversary proceeds in a sequence of activated edges (called rounds), and a protocol can be completely described by the actions of node u (and symmetrically v) during a round $E(u, v)$. Let C denote the size of each node's memory, then Slide+ requires that $C \geq 8n^2$, and for simplicity we will assume that $C/n \in \mathbb{N}$.

During activated edge $E(u, v)$, let $(v, u, (p', h'))$ denote the message that u receives from v in Step 1 of the round (via the scheduling adversary). Also, u has recorded the request $(u, v, (p, h))$ that it made during Step 2 of the previous round in which $E(u, v)$ was activated; note that v will be receiving this message during Step 1 of the current round. Then during round $E(u, v)$, u does:

1. If u is the Sender, then:
   (a) If $h < C$: u deletes packet p from his input stream $\{p_1, p_2, \ldots\}$, ignores the received p', and proceeds to Step (1c)
   (b) If $h' \geq C$: u keeps p, ignores the received p', and proceeds to Step (1c)
   (c) The Sender finds the next packet $p_i \in \{p_1, p_2, \ldots\}$ that has not been deleted and is not currently an outstanding request already sent to the adversary, and sends the request $(u, v, (p_i, C + \frac{C}{n} - n))$ to the adversary.
2. If u is the Receiver, then u sends the request $(u, v, (\perp, \frac{-C}{n} - 2n + 1))$ to the adversary. Meanwhile, if $p' \neq \perp$, then u stores/outputs p' as a packet successfully received.
3. If u is any internal node, then:
   (a) If $h \geq h' + (C/n + 2n)$: u ignores p', deletes p, updates height $h = h - 1$, and proceeds to Step (3d)
   (b) If $h \leq h' - (C/n + 2n)$: u keeps both p and p', updates height $h = h + 1$, and proceeds to Step (3d)
   (c) If $|h - h'| < C/n + 2n$: u ignores packet p', keeps p, and proceeds to Step (3d)
   (d) Node u finds a packet p'' that it has not already committed in an outstanding request to the adversary, and sends the request $(u, v, (p'', h))$ to the adversary

3.2 Our Protocol

Our protocol extends the Slide+ protocol described above to provide security against the node-controlling adversary by using digital signatures in the following two ways:

1. The sender signs every packet, so that honest nodes do not waste resources on modified or junk packets, and so that packets the receiver gets are unmolested
2. Communication between nodes will be signed by each node. This information will be used later by the sender (if there has been malicious activity) to hold nodes accountable for their actions, and ultimately eliminate corrupt nodes

The routing rules for each internal node are the same as in the Slide+ protocol, except that whenever a node u sends a packet to a neighbor v, there will be four parts to this communication:

(a) The packet itself, i.e. a packet from the sender intended for the receiver
(b) The current height of u, i.e. how many packets u is currently storing
(c) A signature on the communication that u has had with v, as described below
(d) Signatures from other nodes that the sender requested, as described below

The first two parts of each communication are identical to the Slide+ protocol, so it remains to discuss the second two items, which are used for the identification of corrupt nodes. Note that the second two items each consist of a signature on some quantity; for this reason we will require that the bandwidth of each edge is large enough to allow for simultaneous transmission of two signatures (plus the packet itself). The signature that u includes on his communications with v for Item (c) above pertains to the following four items:

Sig1 The total number of packets u has sent to v so far
Sig2 The total number of times the previous packet p that was exchanged between them has crossed the edge $E(u, v)$ (can be more than once)
Sig3 The cumulative difference in u and v's heights, measured from each time u and v exchanged a packet
Sig4 An index representing how many times $E(u, v)$ has been activated, to serve as a time-stamp on the above three items

It remains to explain Item (d) from above, for which it will be useful to first describe from a high-level how our protocol handles malicious activity by corrupt nodes. Since secure routing is impossible if either the sender or receiver is corrupted, we will assume that they remain uncorrupted through the protocol, and they will be responsible for regulation of the network (e.g. identifying and eliminating corrupt nodes). Also, because our definition of security (see Section 2) requires that the receiver gets all of the packets sent by the sender, we will use error-correction and first expand the messages into codewords so that the receiver can reconstruct each message if he has a constant fraction of the codeword packets. See e.g. [7] for a specific description of how this can be done. We note that because the definition of throughput only cares about asymptotic performance (i.e. constants are absorbed in the k that appears in Definition 1), the use of error-correction will not affect the throughput of our protocol.

From a high-level, the protocol attempts to transfer one message (codeword), consisting of $\Theta(nC)$ bits, at a time (this is called a message/codeword transmission). The sender will continue inserting packets corresponding to the same codeword until one of the following occurs:

- S1 Sender gets a message indicating the receiver decoded the current codeword
- F2 Sender gets a message alerting him of inconsistencies in height differences
- F3 Sender has inserted all packets corresponding to the current codeword
- F4 Sender gets a message indicating the receiver got the same packet twice
- F5 Sender is able to identify a corrupt node

Cases S1, F2, and F4 come from messages sent by the receiver, a process we do not explicitly describe due to space constraints. In the case of S1 (successful codeword transmission), the sender will begin inserting packets corresponding to the next codeword. In the case of F5, the sender will eliminate the identified node (i.e. alert all nodes in the network to forever ignore the corrupt node), and begin anew transmitting packets corresponding to the current codeword. The other three cases correspond to failed attempts to transfer the current codeword due to corrupt nodes disobeying protocol rules, and in each case the sender will use the signed information from Item (c) above to identify a corrupt node.

In cases F2-F4, the sender will begin anew transmitting packets corresponding to the current codeword. Before nodes are allowed to participate in transferring the codeword packets, they must first learn that the last transmission failed, the reason for failure (F2-F4), and the sender must receive all of *the signatures the node was storing from its neighbors*, called the node's status report (i.e. all signed information from Item (c) above). Note that the network itself is the only medium of communication available for relaying the signatures a node is storing to the sender, and hence part of the bandwidth of each edge (and part of the storage capacity of each node) is devoted to returning these pieces of signed information to the sender (this is Item (d) from the above list).

Until the sender has received a node's status report for a failed transmission, the node will remain on the **blacklist**. That is, no honest node u will transfer any codeword packets to another node v until u obtains verification from the sender that the sender has received v's status report.

4 Analysis of Our Protocol

We use competitive analysis to evaluate the throughput performance of the above protocol. To this end, let $(\mathcal{A}, \mathcal{P}')$ denote an adversary/off-line protocol pair for which we compare our routing protocol $\mathcal{P}$. Due to space constraints, we provide only a proof sketch of Theorem 3 (see [18] for details):

Theorem 3. *If at any time $\mathcal{P}'$ has received $\Theta(xn)$ messages, then $\mathcal{P}$ has received $\Omega((x - n^2))$ messages. Thus, if the number of messages $x \in \Omega(n^2)$, then our protocol has competitive ratio $1/n$.*

Theorem 3 will follow immediately from the following three Lemmas:

Lemma 1. *Suppose transmission* T *failed and at some later time (after* T *but before any nodes are eliminated) the sender has the status report from all nodes not on the blacklist during* T*. Then the sender can eliminate a corrupt node.*

Proof. (Sketch) We split the proof into the three cases of transmission failure:

Handling Case F2. If a misbehaving node u tries to jam the network by duplicating packets, then there will be a discrepancy between the recorded values of cumulative height differences from packets transferred from u's neighbors *to* u and packets transferred *from* u to its neighbors. Therefore, if the sender has Sig3 from all of u's neighbors, the he can identify u as corrupt.

Handling Case F3. The number of packets per codeword ($\Theta(nC)$) is chosen so that even if nC packets are missing, the receiver can still decode. Therefore, since the capacity of the network is bounded by nC, if the sender has inserted all of the codeword packets and the receiver cannot decode, then necessarily a corrupt node is deleting packets. The information from the status reports (in particular information from Sig1) can be used to identify such a node.

Handling Case F4. A corrupt node is duplicating packets, and the status reports (in particular information from Sig2) can be used to identify such a node.

Lemma 2. *There can be at most $n-1$ (not necessarily consecutive) failed codeword transmissions $\{\mathtt{T}_i\}_{i=1}^{n-1}$ (i.e. cases F2-F4) before there is necessarily some $\mathtt{T}_i$ such that the sender has gathered the complete **status report** from every node that was not on the blacklist during $\mathtt{T}_i$.*

Proof. (Sketch) A node is not allowed to participate in a transmission (it is placed on the blacklist) until the sender has received the node's status report(s) for all previous failed transmissions. Thus, for each failed transmission $\mathtt{T}_i$ for which the sender has not collected all status reports, there is (at least) one *distinct* node whose status report is missing, and hence this node will be on the blacklist for all later transmissions until the sender gets this report. Since there are $n-1$ nodes (excluding the sender), this can happen at most $n-1$ times.

The final lemma guarantees that one of the cases S1-F5 necessarily happens by the time the off-line protocol $\mathcal{P}'$ has delivered $O(n^2C)$ packets. For any transmission $\mathtt{T}$, let $Y_{\mathtt{T}}^{\mathcal{P}}$ and $Z_{\mathtt{T}}^{\mathcal{P}}$ (respectively $Y_{\mathtt{T}}^{\mathcal{P}'}$ and $Z_{\mathtt{T}}^{\mathcal{P}'}$) denote the set of packets sent and received during $\mathtt{T}$ by protocol $\mathcal{P}$ (respectively by $\mathcal{P}'$).

Lemma 3. *In any transmission $\mathtt{T}$, $|Z_{\mathtt{T}}^{\mathcal{P}'}| = O(n^2C)$. If the transmission was successful (as in case S1), then $|Z_{\mathtt{T}}^{\mathcal{P}}| = \Theta(nC)$.*

Proof. (Sketch) The second statement of the theorem is immediate, since the receiver requires $\Theta(nC)$ packets to decode a codeword. For the first statement, we first fix a transmission $\mathtt{T}$, and split the packets of $Z^{\mathcal{P}'}$ (we will suppress subscripts $\mathtt{T}$) into subsets as follows. We can view the scheduling adversary $\mathcal{A}$ as simply a schedule (order) of edges that the adversary will honor. We will imagine a *virtual* world, in which $\mathcal{P}$ and $\mathcal{P}'$ are run simultaneously. Let $Z_3^{\mathcal{P}'}$ denote the set of packets in $Z^{\mathcal{P}'}$ that traveled between two ***honest*** nodes in some round of $\mathtt{T}$, and $\mathcal{P}$ did not transfer a packet in this round because (at least) one of these nodes was on the blacklist. Define $Z_1^{\mathcal{P}'}$ to be the subset of $Z^{\mathcal{P}'} \setminus Z_3^{\mathcal{P}'}$ consisting of packets p' for which there exists at least one round $E(u,v)$ such that both p' *and* some packet $p \in Y^{\mathcal{P}}$ were both transferred this round.[4] Set $Z_2^{\mathcal{P}'} = Z^{\mathcal{P}'} \setminus (Z_1^{\mathcal{P}'} \cup Z_3^{\mathcal{P}'})$. Also, let $T^{\mathcal{P}}$ denote the number of packet transfers in $\mathcal{P}$ between two ***honest*** nodes during $\mathtt{T}$. Then Lemma 3 follows from:

Claim. (1) $T^{\mathcal{P}}{=}O(n^2C)$, (2) $|Z_1^{\mathcal{P}'}| \leq T^{\mathcal{P}}_,$ (3) $|Z_2^{\mathcal{P}'}| \leq T^{\mathcal{P}}_,$ (4) $|Z_3^{\mathcal{P}'}|{=}O(n^2C)$

Proof. (Sketch) (1) follows from transfer rules: a packet transfer corresponds to a packet dropping in height by C/n, so this can happen n times per inserted packet, and there are $\Theta(nC)$ packets per codeword. (2) is immediate, and (4) comes from the fact that by the time $O(nC)$ packets have reached any honest node u, the sender will necessarily have received any outstanding status report of u. (3) is the most difficult to obtain, and is done using potential function arguments together with the following observations: 1) when any packet $p' \in Z_2^{\mathcal{P}'}$ is first inserted, it was necessarily inserted into some node u such that *with respect to* $\mathcal{P}$, u had height $\approx C$ (otherwise $\mathcal{P}$ would have also inserted a packet this round, and then $p' \in Z_1^{\mathcal{P}'}$). Similarly, when $p' \in Z_2^{\mathcal{P}'}$ is received by the Receiver from some node v, then *with respect to* $\mathcal{P}$, v had height zero. The idea will then be to assign a potential function $\varphi_{p'}$ to every packet $p' \in Z_2^{\mathcal{P}'}$ that represents the current height *with respect to* $\mathcal{P}$ of the node in which p' is currently stored. Thus, when a packet $p' \in Z_2^{\mathcal{P}'}$ is first inserted, $\varphi_{p'} = C$, and when $p' \in Z_2^{\mathcal{P}'}$ is received, $\varphi_{p'} = 0$. Then roughly speaking, we show that for each $p' \in Z_2^{\mathcal{P}'}$, decreases in $\varphi_{p'}$ can be linked to decreases in height of packets transferred by $\mathcal{P}$.

References

1. Afek, Y., Awerbuch, B., Gafni, E., Mansour, Y., Rosen, A., Shavit, N.: Slide– The Key to Poly. End-to-End Communication. J. of Algo's 22, 158–186 (1997)
2. Afek, Y., Gafni, E.: End-to-End Communication in Unreliable Networks. In: PODC (1988)
3. Afek, Y., Gafni, E., Rosén, A.: The Slide Mechanism with Applications in Dynamic Networks. In: Proc. 11th ACM SODA, pp. 35–46 (1992)
4. Aiello, W., Kushilevitz, E., Ostrovsky, R., Rosén, A.: Adaptive Packet Routing For Bursty Adversarial Traffic. J. Comput. Syst. Sci. 60(3), 482–509 (2000)
5. Aiello, W., Ostrovsky, R., Kushilevitz, E., Rosén, A.: Dynamic Routing on Networks with Fixed-Size Buffers. In: Proc. 14th ACM SODA, pp. 771–780 (2003)
6. Ajtai, M., Aspnes, J., Dwork, C., Waarts, O.: A Theory of Competitive Analysis for Distributed Algorithms. In: Proc. 35th IEEE FOCS, pp. 32–40 (1994)
7. Amir, Y., Bunn, P., Ostrovsky, R.: Authenticated Adversarial Routing. In: 6th Theory of Crypt. Conf., pp. 163–182 (2009)
8. Andrews, M., Awerbuch, B., Fernández, A., Kleinberg, J., Leighton, T., Liu, Z.: Universal Stability Results for Greedy Contention-Resolution Protocols. In: Proc. 37th IEEE FOCS, pp. 380–389 (1996)
9. Awerbuch, B., Azar, Y., Plotkin, S.: Throughput-Competitive On-Line Routing. In: Proc. 34th IEEE FOCS, pp. 401–411 (1993)
10. Awerbuch, B., Holmer, D., Nina-Rotaru, C., Rubens, H.: An On-Demand Secure Routing Protocol Resilient to Byzantine Failures. In: Proc. of 2002 Workshop on Wireless Security, pp. 21–30 (2002)

[4] Note that we make no condition that the two packets traveled in the same direction.

11. Awerbuch, B., Leighton, T.: Improved Approximation Algorithms for the Multi-Commodity Flow Problem and Local Competitive Routing in Dynamic Networks. In: Proc. 26th ACM STOC, pp. 487–496 (1994)
12. Awerbuch, B., Mansour, Y., Shavit, N.: End-to-End Communication With Polynomial Overhead. In: Proc. of the 30th IEEE FOCS, pp. 358-363 (1989)
13. Barak, B., Goldberg, S., Xiao, D.: Protocols and Lower Bounds for Failure Localization in the Internet. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 341–360. Springer, Heidelberg (2008)
14. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Camb. Univ Press, Cambridge (1998)
15. Borodin, A., Kleinberg, J., Raghavan, P., Sudan, M., Williamson, D.: Adversarial Queuing Theory. In: Proc. 28th ACM STOC, pp. 376–385 (1996)
16. Broder, A., Frieze, A., Upfal, E.: A General Approach to Dynamic Packet Routing with Bounded Buffers. In: Proc. 37th IEEE FOCS, pp. 390–399 (1996)
17. Bunn, P., Ostrovsky, R.: Throughput in Asynchronous Networks. arXiv Technical Report, arXiv:0910.4572 (2009)
18. Bunn, P., Ostrovsky, R.: Asynchronous Throughput Optimal Routing in Malicious Networks. IACR Eprint Archive, Report 2010/231. April 2010 (2010)
19. Goldberg, S., Xiao, D., Tromer, E., Barak, B., Rexford, J.: Path-Quality Monitoring in the Presence of Adversaries. SIGMETRICS 36, 193–204 (2008)
20. Kushilevitz, E., Ostrovsky, R., Rosén, A.: Log-Space Polynomial End-to-End Communication. SIAM Journal of Computing 27(6), 1531–1549 (1998)
21. Leighton, T., Makedon, F., Plotkin, S., Stein, C., Tardos, É., Tragoudas, S.: Fast Approximation Algorithms for Multicommodity Flow Problem. In: Proc. 23rd ACM STOC, pp. 101–111 (1991)
22. Sleator, D., Tarjan, R.: Amortized Efficiency of List Update and Paging Rules. Commun. ACM 28(2), 202–208 (1985)