ClariNet: Generalized Reputation Assessment for Witnessed Data Exchange

***Abstract*—Audit logging, a necessity for post-incident analysis, can break down when two participants—such as nodes in a distributed system communicating over a network—have different records, and neither is able to definitively prove its claim. Introducing a witness that records data in transit can provide a tie-breaking vote, but requires the witness be honest. In this paper, we present ClariNet, a reputation assessment scheme that allows participants in a distributed system to identify activity indicative of audit log forgery, penalize such malicious activity even when the exact source is not known, and ensures that the true origin of the malicious activity is in aggregate penalized more harshly than cooperative participants. It does this by verifying claims against a known point of reference, providing verifiable claims to peers, and introducing the notion of different penalty degrees. We demonstrate through formal proofs and simulation that ClariNet allows cooperative participants to gain meaningful insight into peer behavior even when over half the network is malicious and when malicious peers act maliciously relatively infrequently (as low as 10%). With ClariNet, participants in distributed systems can gain confidence in identifying a cooperative witness and protect themselves from false claims.**

I. Introduction

In distributed systems, participants must often send data to one another. These participants typically keep audit logs to facilitate post-incident analysis. Investigators can use these audit logs to determine data flow and identify corresponding actions, allowing them to identify where expectation diverges from reality. Identifying these divergences can greatly expedite root causes identification which is becoming ever more important as systems grow in scale and complexity.

However, parties must ensure that audit logs are responsibly recorded and maintained. Missing logs provide no benefit, and inaccurate logs can result in false leads and wasted time. Malicious participants can exacerbate these problems by intentionally recording data that attempts to shift suspicion. While the other party may likely keep its own logs, unless the behaving participant has some means of supporting its claim, these conflicting logs can lead to further wasted time arguing over which account is correct. While cryptographic signatures can assist in identifying the origin of a message, a malicious party could provide an invalid signature undermining this proof of origin.

Consider an IoT security system composed of cameras and hubs that can contact a security service. The cameras and hubs are manufactured by different companies and equipped with onboard AI models to facilitate capabilities such as image cleanup and recognition. Unnecessarily notifying the security service is not ideal, but is significantly less harmful than failing to notify of a real security event, so the hubs are permitted to make decisions even when the origin of the event message is not verifiable.

A malicious participant could exploit this, such as if it lacks faith in its cleanup. In one scenario, the camera cleans an image and supplies the cleaned image and an invalid signature while logging the uncleaned image and a valid signature. In the other, the hub receives an uncleaned image and a valid signature but records its own cleaned image and an invalid signature as if these were what the camera transmitted. Despite different bad actors, both scenarios result in the same disagreeing logs.

Having an intermediary, such as another smart device, *witness* the data in transit can point investigators in a particular direction by supporting one participant's account. This requires the witness be impartial and responsibly record data. In a system with no agreed-upon trusted party, participants must rely on some means of identifying trustworthy witnesses, such as a *reputation* scheme.

To this end, we propose ClariNet, a protocol participants can use to identify and penalize malicious actions that facilitate log forgery. In addition to identification, ClariNet resists a malicious intermediary poisoning the communication between two behaving participants. ClariNet accomplishes this by providing three reputation actions—*rewards* and *weak* and *strong penalties*—application rules, correctness criteria, and rules for safely sharing audit information. ClariNet uses a combination of a known point of reference and obvious signs of malicious activity—invalid signatures—to identify malicious action and applies rewards when peers behave, weak penalties when the malicious party is unclear, and strong penalties when the malicious party is certain.

We evaluate these rules using formal proof and simulation using PeerSim [29] to demonstrate the following:

- Participants always end up with either irrefutable proof a peer sent, witnessed, or received a message via signatures, or are able to penalize bad actors.
- For a given message, a participant never penalizes behaving peers more harshly than malicious peers.
- For a given message, the aggregate penalties for malicious peer(s) always outweigh the incorrect penalties for behaving peer(s).
- Malicious participants' average reputations decrease more quickly than cooperative participants'.

Through the simulations, we demonstrate *(4)* holds under a variety of network conditions such as frequency of malicious action and proportion of malicious peers.

Altogether, we make the following contributions:

- ClariNet: a novel reputation assessment system that can differentiate known origins of malicious action from suspected origins and resists malicious exploitation.
- Formal proofs demonstrating ClariNet's soundness.
- Simulations demonstrating that ClariNet's theoretical claims hold at scale.

We begin by discussing system requirements and then discuss the threat model. Next, we provide ClariNet's behaviors followed by formal proofs. Then simulation setup and results. We close with limitations, related work, and conclusion.

II. Problem Statement

Audit logging is a necessity for post-incident analysis, particularly when two systems have communicated and auditors must determine a responsible party. In the absence of centralized, trusted logging, auditors must rely on participant logs which can be forged by a malicious participant. When each participant claims a different log record, it can degenerate into one party's word against the other.

Abstract example. P_A and P_B are participants, D and D' represent data such that $D \neq D'$, S is a valid signature, and I an invalid signature.

Scenario 1. P_A sends a pair of data D and signature S, i.e. (D, S), and logs it. P_B receives the pair, but it does not want to log the receipt of D. In response, P_B invents new data D' and proceeds to act on it. Since P_B cannot generate S for P_A, it logs (D', I).

Scenario 2. P_A has data D' it should send based on some agreement, but wishes to deny it sent D', so it sends (D', I) and logs (D, S) where D is the data it wishes to claim. P_B receives (D', I), takes appropriate action, and logs (D', I).

In both scenarios, P_A claims it sent (D, S) while P_B claims it received (D', I). In scenario 1, P_B is the malicious actor, while in scenario 2, P_A is the malicious actor. Therefore, given only the logs, it is impossible to tell who is the bad actor.

III. ClariNet: Witnesses and Reputation

In ClariNet, a witness records the message in transit and serves as a tie-breaking vote, but this requires the witness be impartial. The sender and receiver would like some insight into a given witness's trustworthiness, which they can achieve through a *reputation*. They would then use this reputation during a witness agreement process to find a witness they deem acceptable. To do this, they need some criteria by which to form that reputation. These criteria must also be resistant to a malicious witness that simply wishes to poison the communication between benign senders and receivers.

To accomplish this, participants need correctness criteria, reputation adjustment rules, information exchange rules, and a secure means of authenticating their peers. In designing ClariNet, we address each in the face of intelligent malicious adversaries who are aware of ClariNet.

IV. Threat Model

Adversaries are any participants that wish to avoid culpability for data they sent or received. To accomplish this, they need to achieve three goals:

G1. If sending, provide the messages in a deniable fashion.
G2. Provide proof of messages supporting the forged claim.
G3. Win the consensus protocol.

G1 and *G2* are reasonably simple. Providing an invalid signature on the delivered message ensures the receiver does not have definite proof the adversary sent the message. Similarly, recording the desired message accomplishes *G2*.

For *G3*, the adversary must find a peer willing to support its false claim. While we leave witness agreement flexible, such agreements would incorporate reputation scores. This means that adversaries want to degrade non-colluding peers' reputations of each other while keeping the adversary and its colluders' reputations high.

A. Assumptions

AS1. Adversaries are capable of intercepting and rewriting messages, but do not wish to entirely halt communication between cooperative participants.
AS2. Adversaries recognize their colluders and can communicate out of band, including sharing private keys.
AS3. Adversaries control what they report to their peers.
AS4. Adversaries make an honest effort to deliver data, just in a manipulated fashion when they deem appropriate.

The qualification to *AS1* is necessary to make the problem tractable. Without it, adversaries could halt all communication between non-colluding peers other than the small subset they deem acceptable. While traditional Byzantine solutions mitigate this with deadlines and default actions, we believe this weakening assumption is acceptable because adversaries still wish to gain the benefits of the distributed system. Halting all communication would reduce the system's capability and require the adversaries to take on more work. Instead, adversaries only interfere when they have strong reason to suspect it would benefit them.

AS2 simply acknowledges that out-of-band communication is possible and cooperative participants cannot rely on intercepting adversarial coordination.

AS3 acknowledges that participants cannot know the internal state of their peers beyond what the peers report, that these reports are at the peers' discretion, and can be false.

AS4 means that should an adversary have message M that it should deliver to a recipient, it delivers M, though M may be manipulated. In conjunction with *AS1*, this means that adversaries do not prevent a cooperative witness from delivering a message to the receiver. This is a simplifying assumption and a potential area for future work.

V. ClariNet Specification

A. Participant Identity and Authentication

Without identity and authentication, participants have no way of verifying the peer with whom they are communicating. They can accomplish this with public key cryptography and tying identity to public keys, similar to libp2p [30]. By combining this with authenticated key exchange (AKE) [31], participants can exchange public keys ad hoc, identify the peer by hashing the public key, and then confirm that the peer is who it claims to be by using the public key as part of an AKE handshake with an additional ephemeral key. The handshake can only succeed if the peer possesses the private key corresponding to the shared public key that itself corresponds to the peer's claimed identity, which we call ID_P. From this point on, participants can be sure that messages encrypted with this session key come from the expected peer. These guarantees hold as long as participants use strong keys and prevent key compromise. Libp2p uses the Noise Protocol Framework [32] [33] to implement this for real-world systems such as the Interplanetary File System [34].

While this ensures that peers cannot impersonate each other, it does not solve changing identity or allow participants to authenticate peers on a higher level. These are out of scope and future work could incorporate mitigations.

B. Connections

We introduce logical *connections* which are a record of the sender S, receiver R, and witness W for some set of messages they wish to send. This is necessary so the participants, i.e., S, R, and W, know which of their peers should be considered when sending or receiving a particular data message. We do this as witness selection may be expensive and participants would like to have a stable witness for multiple messages. A connection is uniquely identified with some separate identifier that all participants agree upon. We refer to this identifier as ID_C. We require only that ID_C is unique, $S \neq W \wedge R \neq W$, and S, W, and R are fixed. This means that any participant changes necessitate a new connection, and that simultaneous connections may exist containing the same participants in the same positions, but with a different ID_C. Messages are uniquely identified by the ID_C and a sequence number $SeqNo$. We refer to this combination as ID_M.

We leave the process of connection initiation and termination intentionally flexible to prevent premature calcification.

C. Correctness Criteria

The correctness goals are to detect when an adversary may be attempting to deny a message or harm a peer's reputation. To detect these, we use two criteria:

CC1. Invalid signatures
CC2. Known points of reference

The formal state machines in Section VI demonstrate how each is used to allow participants insight into their peers.

D. Messaging

We require only that underlying transport be reliable such as TCP or QUIC [35]. This ensures that both sides are confident the message is delivered without error.

ClariNet has seven operations, Op1-Op7, that fall into two categories.

- Data Delivery
  - Op1. Sending
  - Op2. Witness Sending
  - Op3. Receiving
- Auditing
  - Op4. Receiver Forward Receiving
  - Op5. Querying
  - Op6. Query Answering
  - Op7. Query Forward Receiving

Connections only apply to *data delivery*. *Auditing* utilizes knowledge from connections, but does not require an active one. While *Op4* is an audit operation, it occurs as part of data delivery. We require signatures on all messages and assume all participants log all messages. Fig. 1 shows message direction for all operations.

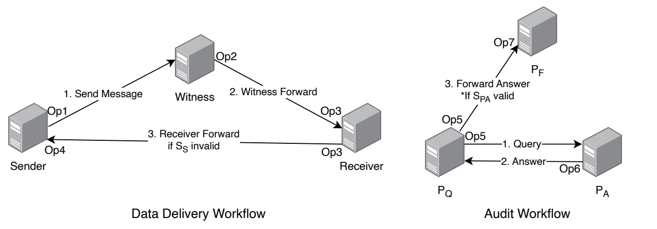

Fig. 1. Message Operations in ClariNet.

1) Auditing: While data delivery provides insight for incoming communications, it does not provide any for outgoing. This is why auditing is necessary. Any participant may initiate an audit at any time for any message. We discuss only querying for messages from connections participants were members of, but we do not forbid querying for other messages.

Auditing leverages both *CC1* and *CC2* to allow S and W greater insight into outgoing communications. R is also free to query, but likely gains less due to its better visibility during data delivery. Forwarding is necessary to guard against a participant that attempts to claim different messages to different peers.

E. Reputation Operations

X represents the affected peer in the three operations:

- Reward or $rew(X)$,
- Weak penalty or $pen_W(X)$, and
- Strong penalty or $pen_S(X)$.

Rewards are given when a participant observes *correct* behavior from a peer X. S and W do not give rewards during data delivery only to simplify the logic. Rewards must increase X's reputation, i.e. $rew(X) > X$.

Weak penalties are given when a participant observes malicious action that X *may* have caused. They are always given to all known potential sources. Weak penalties must decrease X's reputation, i.e. $pen_W(X) < X$.

Strong penalties are given when a participant observes malicious action it is *sure* X caused. These are only given to a single X. Strong penalties must decrease X's reputation more than weak penalties, i.e. $pen_S(X) < pen_W(X)$.

1) Double Counting: It is important we prevent double counting, otherwise situations might arise where a participant penalizes a malicious peer less harshly than a cooperative peer. For example, a malicious witness W alters the message M between cooperative S and R. If S queries W, W reports M so S rewards W. Then S queries R, receives M', and weakly penalizes both W and R, resulting in a final state of $(rew(W) + pen_W(W)) > pen_W(R)$.

To prevent this, participants must ensure that for every unique (ID_M, ID_P) there exists only one reputation operation and that it is always the harshest observed, with $pen_S > pen_W > rew$. We refer to these records as *assessments*.

VI. Formal State Machines

In this section, we present the ClariNet protocol state machines that formally define the previous section and which will be proved as sound in the next section.

In addition to the terminology used previously, we define:

- $sign_X(Y)$ signs message Y using X's private key.
- $hash(Y)$ returns the hash of Y using the agreed upon hashing algorithm.
- $verify(X, Y, Z)$ returns true if Z is a valid signature for data Y using X's public key.
- $verifyHash(X, Y, Z)$ verifies signature Z for pre-hashed Y such that:
  $verify(X, Y, Z) = verifyHash(X, hash(Y), Z)$.

Additionally, we define direct communication to mean that within a connection the two participants had no intermediary, i.e., S and R do not directly communicate, but all others do.

For *Op5-7*, P_Q is the participant initiating a query, P_A the one being queried, and P_F the third participant in the connection for the queried message.

Unspecified branches are left intentionally blank to allow flexibility and prevent premature calcification.

A. *Op1: Sending*

```
1) Construct $ID_M = (ID_C, SeqNo)$
2) Construct $S_S = sign_S(ID_M, D)$
3) Deliver $(ID_M, D, S_S)$ to witness $W$
```

B. *Op2: Witness Sending*

```
1) Receive $(ID_M, D, S_S)$ from a participant $P$
2) Check $P = S$
       F $pen_S(P)$ and halt
3) Check corresponding open connection
       T Continue
4) $verify(S, (ID_M, D), S_S)$
       F $pen_S(S)$
5) Sign $S_W = sign_W(ID_M, D, S_S)$
6) Deliver $(ID_M, D, S_S, S_W)$ to receiver $R$
```

C. *Op3: Receiving*

```
1) Receive $(ID_M, D, S_S, S_W)$ from $P$
2) Check $P = W$
       F $pen_S(P)$ and halt
3) Check corresponding open connection
       T Continue
4) $verify(W, (ID_M, D, S_S), S_W)$
       F $pen_S(W)$ and halt
5) $verify(S, (ID_M, D), S_S)$
       T $rew(S)$, $rew(W)$, and halt
       F $pen_W(S)$, $pen_W(W)$, and continue
6) Construct $H = hash(ID_M, D, S_S)$
7) Construct $S_R = sign_R(ID_M, H, S_W)$
8) Deliver $(ID_M, H, S_W, S_R)$ to $S$
```

D. *Op4: Receiver Forward Receiving*

```
1) Receive $(ID_M, H, S_W, S_R)$ from $P$
2) Check $P = R$
       F $pen_S(P)$ and halt
3) $verify(R, (ID_M, H, S_W), S_R)$
       F $pen_S(R)$ and halt
4) $verifyHash(W, H, S_W)$
       F $pen_S(R)$ and halt
5) Construct $V = hash(ID_M, D, S_S)$ from $S$'s record
6) Check $V = H$
       F $pen_S(W)$
```

E. *Op5: Querying*

```
1) Select some message $M$ for which to query
2) Construct $S_{P_Q 1} = sign_{P_Q}(M.ID_M)$
3) Deliver $(M.ID_M, S_{P_Q 1})$ to $P_A$
4) Receive $A = (ID_M, H, S_{P_A})$ from $P_A$
5) $verify(P_A, (A.ID_M, A.H), A.S_{P_A})$
       F $pen_S(P_A)$ and halt
6) Construct $V = hash(M.ID_M, M.D, M.S_S)$
7) Check $V = M.H$
       T $rew(P_A)$
       F Check direct communication between $P_Q$ and $P_A$
           T $pen_S(P_A)$
           F $pen_W(P_A)$, $pen_W(P_F)$
8) Optionally halt
9) Construct $S_{P_Q 2} = sign_{P_Q}(A)$
10) Construct $F = (A, ID_{P_A}, S_{P_Q 2})$ where $ID_{P_A}$ is the
    $ID_P$ of $P_A$
11) Deliver $F$ to $P_F$
```

F. *Op6: Query Answering*

```
1) Receive $Q = (ID_M, S_{P_Q})$
2) Check for record of message $M$ corresponding to $ID_M$
       T Construct $H = (M.ID_M, M.D, M.S_S)$
       F Construct $H = null$
3) Sign $S_{P_A} = sign_{P_A}(ID_M, H)$
4) Deliver $(ID_M, H, S_{P_A})$ to $P_Q$
```

G. *Op7: Query Forward Receiving*

```
1) Receive $F = (A, ID_{P_A}, S_{P_Q})$
2) $verify(P_Q, (A, ID_{P_A}), S_{P_Q})$
       T $pen_S(P_Q)$ and halt
3) $verify(P_A, (ID_M, H), S_{P_A})$
       F $pen_S(P_Q)$ and halt
4) Find record of message $M$ corresponding to $ID_M$
5) Construct $V = hash(M.ID_M, M.D, M.S_S)$
6) Check $V = H$
       T $rew(P_A)$
       F Check direct communication between $P_F$ and $P_A$
           T $pen_S(P_A)$
           F $pen_W(P_A)$, $pen_W(P_Q)$
```

VII. Analysis

We focus only on cooperative participants because ClarINet's goal is to allow cooperative participants insight into which peers may be malicious. Malicious participants' actions only matter in how they affect cooperative participants.

We first define the potential outcomes of an interaction for each connection participant. This includes the messages they may see, the reputation action, and an explanation of why this is desirable. We then define the different potential combinations of participants in a connection and use the outcomes to demonstrate how the protocol invariants hold.

A. Data Delivery

1) Sender S: S's only insight during data delivery is if the receiver R forwards a message. First we define some ground rules to limit the number of possible values for the forward's fields.

Lemma 1. The hash H that S receives from R has only two states: correct and incorrect.

Proof. Because S has a record of the components of H, i.e. ID_M, D, and S_S, it can generate a baseline hash, V, for comparison. Since comparison of H and V is simple boolean equality, only two possibilities exist: $true$ and $false$. Because correctness is a direct mapping from the boolean outcomes, $true \rightarrow correct$ and $false \rightarrow incorrect$, only two outcomes exist for verifying H. □

Lemma 2. H suffices for S to know the information R claims to have received was modified.

Proof. By definition, $H = hash(ID_M, D, S_S)$. Provided the hashing algorithm is robust, $H = H'$ *iff* the components used to generate H' are identical to those used to generate H. By *Lemma 1*, S can know what value H should have and use this to verify H. Because $H = H'$ *iff* the components are identical, if H is incorrect, S knows at least one of ID_M, D, or S_S does not match what it sent. □

While H does not include S_W, it has no bearing on what S sent or whether R received the message unmodified. Therefore, there is no need for S to verify S_W. This is beneficial because S cannot have a baseline for S_W without additional networking overhead. S_W and S_R have only two states each due to being cryptographic signatures. Together, these yield the 8 possibilities. Table I lists the realistic ones. Omitted permutations would result in S strongly penalizing R so neither a cooperative nor a malicious R would ever perform these and a malicious W would have no control over them.

Lemma 3. Given a message forward containing an invalid H and a valid S_W, S can be sure W is the source of discrepancy.

Proof.
1) R cannot generate a valid S_W without W's private key, which we assume W does not leak.
2) By protocol definition, the same hash algorithm is used to generate both S_W and H.
3) By protocol definition, R generates H from the message it received.
4) By (1), R cannot generate a valid S_W should it attempt to generate H using any different data.
5) Therefore, if S_W is valid, S can safely assume that R generated H from exactly the data W delivered.
6) Therefore, S can safely assume that W held modified data while generating S_W.
7) By protocol definition, S and W directly communicate.
8) Therefore, no intermediary could have modified the data between S and W.
9) Therefore, W must be the one who modified the data resulting in H and S_W.

□

2) Witness W: W can only assess based on S_S, shown in table II.

3) Receiver R: R can assess based on S_S and S_W for the four outcomes in table III. In **RS2** and **RS4**, R does not alter S's reputation. We do this for protocol simplicity—an invalid S_W means R always halts assessment and can make assessments during a later query. Additionally, it introduces some conservativism in penalties because R cannot forward in **RS4**.

B. Auditing

All participants follow the same auditing behavior; only penalties differ based on direct communication. Because of this similarity, we separate some of the reputation operations. We outline the query outcomes in Table IV, the forward interactions in Table V, and the additional penalty operations in Table VI. In VI, O represents the other participant in the communication. In V, we omit scenarios that P_Q would never perform since they would result in P_F strongly penalizing it to no benefit.

C. Scenarios

A connection has three participants which can each be cooperative or malicious for eight possible configurations.

The malicious participants may either be colluding or acting independently. For this analysis, we assume all malicious participants are colluding because malicious participants benefit from eliminating non-colluding malicious peers.

Lemma 4. Malicious participants benefit from eliminating non-colluding malicious peers from other peers' witness candidate pools.

Proof. A malicious participant P_M wishes to exploit the consensus protocol to win a claim over the other participant in the communication. To do this, it must have its own record and a witness W supporting its claim. While any two form the necessary consensus, a dispute would not occur if S and R agree because W only logs the message without taking action.

P_M also wishes to ensure that W's false report matches. If W reports correctly, P_M loses the dispute via the consensus protocol, and if W false reports differently, either P_M still loses if W agrees with the other side or all three participants have different answers. If all three differ, none win, and all three likely receive additional scrutiny, which malicious

TABLE I
S SEND INTERACTIONS

Outcome ID	H	S_W	S_R	Action	Action Reason	Malicious Participant Reason
SS1	Incorrect	Valid	Valid	$pen_S(W)$	S knows W altered M by *Lemma 3*	Malicious W is attempting to turn S and R against each other

TABLE II
W SEND INTERACTIONS

Outcome ID	S_S	Action	Action Reason	Malicious Participant Reason
WS1	Valid	No action	W has proof S sent M	Malicious S knows it won't want to later deny M
WS2	Invalid	$pen_S(S)$	W knows S violated protocol	Malicious S wishes to later deny having sent M

TABLE III
R SEND INTERACTIONS

Outcome ID	S_S	S_W	Action	Action Reason	Malicious Participant Reason
RS1	Valid	Valid	rew(S), rew(W)	R has proof S sent M and W witnessed M	Malicious S and/or W know they won't want to deny the message
RS2	Valid	Invalid	$pen_S(W)$	R knows W violated the protocol	Wouldn't occur because no benefit
RS3	Invalid	Valid	$pen_W(S)$, $pen_W(W)$	R knows malicious action occurred, but cannot be sure if S or W is source	Malicious S wants to later deny M or malicious W wants to turn S and R against each other
RS4	Invalid	Invalid	$pen_S(W)$	R knows W violated the protocol	Malicious S and W want to deny sending and witnessing M or W_M wants to prevent R from forwarding M

TABLE IV
QUERY INTERACTIONS

Outcome ID	H	S_T	Action	Action Reason	Response Reason
Q1	Correct	Valid	$rew(P_A)$	P_Q has proof P_A sent, witnessed, or received M	P_A is cooperative or malicious P_A does not wish to deny M
Q2	Correct	Invalid	$pen_S(P_A)$	P_Q knows P_A violated the protocol	Malicious P_A wishes to prevent forwarding
Q3	Incorrect	Valid	See Table VI	See Table VI	Malicious P_A wishes to claim alternate message or cooperative P_A received modified message
Q4	Incorrect	Invalid	$pen_S(P_A)$	P_Q knows P_A violated protocol	Malicious P_A wishes to claim an alternate message while preventing forwarding

participants wish to avoid as it may reveal their malicious activities.

Because CLARINET does not require a specific witness agreement algorithm, P_M may not be able to force a colluding W. It thus benefits from making non-colluders, both cooperative and malicious, appear less trustworthy than colluders. □

Lemma 5. Malicious participants benefit from retaining their own record of peers' reputations for non-colluding peers.

Proof. Following from *Lemma 4*, since malicious participants wish to assist peers in removing non-colluding malicious peers, it is trivial for them to track non-colluding peers who may be malicious. This allows them to influence witness agreement toward cooperative peers when they cannot steer it toward one of their colluders.

Thus, it is in their best interest to follow the protocol regarding non-colluding malicious peers. □

By Lemmas 4 and 5, we can simplify the scenarios so that when two malicious participants are present, they are colluding since one that isn't colluding behaves cooperatively. In addition, the primary CLARINET invariants center around always penalizing malicious action and more harshly penalizing the true malicious actor. Because of these, we exclude the scenarios involving two or more malicious participants because there is no risk of accidentally penalizing cooperative peers. We also exclude the scenario where all participants are cooperative since, by definition, no malicious action occurs. This gives the three scenarios in table VII.

It might seem that a malicious participant would not act maliciously when it knows W is not a colluder, but a particularly aggressive malicious participant may still behave maliciously and attempt to claim that W and the other participant are malicious colluders. Table VIII shows penalties for all scenarios once all cooperative participants have finished auditing; $rew = 1/1$, $pen_W = 0/1$, and $pen_S = 0/3$.

D. Scenario (1) - S: Coop, W: Coop, R: Mal

By Section VII-A, the malicious receiver R never forwards a message, and because both S and W are cooperative, both always provide valid signatures. This means no penalties occur during data delivery. When querying, S and W always receive the correct data from each other, so they initially reward each

TABLE V
QUERY FORWARD INTERACTIONS

Outcome ID	H	S_T	S_Q	Action	Action Reason	Response/Forward Reason
F1	Correct	Valid	Valid	$rew(P_A)$	P_F has proof P_A sent, witnessed, or received M	P_A is cooperative or malicious P_A does not wish to deny M
F2	Incorrect	Valid	Valid	See Table VI	See Table VI	Malicious P_A wishes to claim alternate message or cooperative P_A received modified message

TABLE VI
QUERY/FORWARD PENALTIES

Communication	Action	Reason
Direct	$pen_S(P_A)$	P_Q or P_F can be sure P_A is the source of malicious action
Indirect	$pen_W(P_A)$, $pen_W(O)$	P_Q or P_F knows malicious action occurred, but cannot be sure if P_A or O is source.

TABLE VII
CONNECTION PARTICIPANT POSSIBILITIES

Scenario No.	Sender	Witness	Receiver
(1)	Cooperative	Cooperative	Malicious
(2)	Cooperative	Malicious	Cooperative
(3)	Malicious	Cooperative	Cooperative

other, but when either queries R, R may wish to deny it received M.

By Table IV, R's most likely choice is **Q3**. **Q1** undermines its ability to claim a different message and **Q2** and **Q4** cause S to strongly penalize R. **Q3** allows R to maintain its false claim, only be weakly penalized, and even cause S to weakly penalize W. However, because W and R directly communicated, all of **Q2**-**Q4** cause W to strongly penalize R.

E. Scenario (2) - S: Coop, W: Mal, R: Coop

This scenario proves the most problematic because witness W can alter the message to turn R and S against each other and answer both with their expected message. Forwarding mitigates this. During data delivery, R can supply S with definite proof of W's malicious activity. Similarly, when S queries W, W must choose among **Q2**-**Q4**. **Q2** means S can forward the answer to R who would strongly penalize W. **Q3** and **Q4** mean S would strongly penalize W, but W would retain its reputation with R. Regardless of what action W takes, it suffers a strong penalty from S, R, or both and can only cause S and R to weakly penalize each other.

We briefly discuss the sequence of events that might lead to each sub-scenario in Table VIII:

1) W alters M and includes a valid signature S_W, which causes R to forward the message and S to strongly penalize W. Later, S queries R, sees the diverging message, and weakly penalizes R. S queries W sees what it expects, but W keeps its strong penalty. S forwards this answer to R who sees W's valid signature with a different message and strongly penalizes W.

2) W alters M and includes a valid S_W, which causes R to forward the message and S to strongly penalize W. Later, S queries R, sees the diverging message, and weakly penalizes R. S queries W, gets an invalid signature, and can't forward to R.

When R queries W it sees what it expects, but when it queries S, it sees diverging data with S's valid signature so it weakly penalizes S and W.

3) W alters M and includes an invalid S_W, which causes R to strongly penalize W but prevents it from forwarding. Later, S queries R, sees the diverging message, and weakly penalizes W and R. S queries W and sees what it expects.

When R queries W, W includes an invalid signature to prevent R from forwarding and W keeps its strong penalty. When R queries S, it sees diverging data with S's valid signature and weakly penalizes S.

F. Scenario (3) - S: Mal, W: Coop, R: Coop

If S wishes to maintain deniability, it must provide an invalid S_S which would cause W to strongly penalize S and R to weakly penalize W and S since the cooperative W provides a valid S_W. R is unlikely to gain further insight during queries since S most likely uses **Q3** because it knows W already strongly penalized it and can supply an H matching its alternate data. W supplies **Q1** to R so R never does harsher than weakly penalize W. W can reward R because R provides **Q1**.

G. Malicious Interference

In *AS1* we specified that adversaries are capable of intercepting and modifying messages, but that they do not wish to halt all communication between non-colluding peers. Additionally, all communication uses AKE with an ephemeral key to set up a session key and cooperative participants do not leak keys. This means that while malicious participants can intercept messages, they cannot successfully read or modify them. If they attempt to modify the cyphertext, the recipient's decryption will fail which is effectively a corrupted message and would trigger a retransmission of the underlying reliable transport protocol.

Because malicious participants cannot read or successfully modify intercepted messages, they must rely on heuristics such as timing and messaging patterns if they wish to block potentially harmful messages. CLARINET is by design an eventually consistent protocol which helps to mitigates this. Message and answer forwards need only be delivered at some point in the future; their senders are free to attempt this delivery immediately, batch them, or queue them for

TABLE VIII
REPUTATION OPERATION APPLICATIONS

	Scenario (1)			Scenario (2)									Scenario (3)		
				1) W attempts to maintain both			2) W sacrifices reputation with S			3) W sacrifices reputation with R					
	S	W	R	S	W	R	S	W	R	S	W	R	S	W	R
S	-	0/1	0/1	-	0/3	0/1	-	0/1	0/1	-	0/3	0/1	-	-	-
W	1/1	-	0/3	-	-	-	-	-	-	-	-	-	0/3	-	1/1
R	-	-	-	0/1	0/3	-	0/1	0/3	-	0/1	0/1	-	0/1	0/1	-
total	1/1	0/1	0/4	0/1	0/6	0/1	0/1	0/4	0/1	0/1	0/4	0/1	0/4	0/1	1/1

when the sender is otherwise inactive. Additionally, since communication uses reliable transport protocols, participants can be sure that the recipient received the message correctly. Should delivery fail at one time, they can try again later. Because of the variability in these communications, it means malicious participants must take on significant additional state and heuristics may be unreliable, even potentially blocking beneficial messages. For example, a malicious participant may drop something it believes to be an answer forward, but might actually be connection setup, teardown, or a data message. Even if the malicious participant is correct, the answer forward may be for a different, non-colluding malicious peer which, as discussed, the participant would like to be penalized.

1) Key Sharing: AS2 allows malicious participants to share their private keys and even session keys. CLARINET cannot detect this directly, but it does still detect any malicious action that these peers might perform. While it would result in penalizing the wrong peer, only malicious participants share their keys, so even though the participant that shared its key is penalized instead of the malicious actor, a malicious participant is still correctly penalized. Additionally, if a malicious participant is willing to share its key with a peer, it almost certainly means the two are colluding and a member of the collusion group is still correctly penalized.

VIII. PROTOCOL OVERHEAD

While VII demonstrates that CLARINET's invariants hold, a protocol is not practical if it requires excessive overhead. In this section we explore CLARINET's overhead compared to a generic protocol that retains similar logging, just omitting the witness and audit operations. Overhead is divided into two categories: network and storage. Network overhead includes both latency and additional network load. Storage addresses the additional space necessary to store the auditing information.

A. Network

1) Latency: We only discuss latency with regards to data delivery. Auditing can be performed asynchronously and does not have a strict deadline. Data delivery meanwhile presumably has some desired deadline or latency requirements.

This varies significantly based on participant location and network size, but can be generalized to an average of the peers a participant communicates with. We define this average to be L. In the baseline, the latency is just L because S delivers the message directly to R.

If we assume a fully connected network with no particular preference for communication partners[1], this average applies to both the channel between S and W and the channel between W and R. Because data delivery involves both channels, latency is $2L$. While this is not ideal, it should not be acceptable in many situations.

2) Load: Both the hypothetical baseline and CLARINET transmit the same data during data delivery. We measure load in terms of bytes transmitted on the network, so this does not change despite both S and W needing to transmit the data. As such, the data transmitted cancels out and we can focus solely on the additional fields and messages CLARINET requires. Table IX defines some reasonable component implementations and their associated sizes in bytes. We use 64-bit integers because it would allow sending over 9 quintillion ($9e18$) message in a single connection. It would take sending 28 billion messages per second for 10 years to exhaust this.

TABLE IX
PROTOCOL COMPONENT SIZES

Component	Implementation	Size (bytes)
ID_C	UUID	16
Sequence Number	64-bit integer	8
Hash	SHA-256	32
Signature	Encrypted hash	32

Data delivery requires, in addition to the data, an $ID_M = (ID_C, SeqNo)$, S_S, and S_W. The base case also requires S_S, so this cancels. This gives a total overhead of 56 bytes. While S_W is not included in the message S delivers to W we retain it in full to provide an upper bound for the most expensive portion.

In addition to data delivery, there is receiver forwarding. This consists of an ID_M, a hash, S_W, and S_R for a total of 120 bytes. In total, data delivery entails an overhead of $56 + 120 = 176$ bytes.

Auditing a message consists of an initial query, an answer, and an answer forward per participant per peer involved in the connection. Since there are always 3 participants in a

[1]We acknowledge this is unrealistic for real-world networks, but for abstract analysis this should suffice.

connection, each participant would query once per peer for a total of two queries. While participants could query more than once per peer, they would not gain any benefit since it is unlikely a peer would change its reporting. This yields 6 total queries (2 queries per each of 3 participants) at max. This may be lower, for example if a participant does not need to query a peer because it received a fully informative answer forward, but we consider it in full to provide an upper bound.

A query consists of ID_M and a signature totaling 56 bytes. An answer consists of ID_M, a hash, and a signature totaling 88 bytes. An answer forward consists of the 88-byte answer, P_A's ID_P, and the forwarder's signature. ID_P is the hash of the public key and is therefore also 32 bytes, so the answer forward totals 152 bytes. In total auditing entails 296 bytes for a total of $1,776$ bytes across all 6 queries.

In total CLARINET requires 1,952 bytes of overhead per message. We collect these numbers in table X. While this is not completely negligible, it compares reasonably to commonly deployed tools like HTTP headers. The SPDY whitepaper [36] found that HTTP headers range from 200 bytes to 2KB with typical cases being 700-800 bytes for modern sites.

TABLE X
MESSAGE SIZES

Message	Components	Overhead (bytes)
Send	(ID_M, D, S_S, S_W)	56
Receiver Forward	(ID_M, H, S_W, S_R)	120
Query	(ID_M, S_{P_Q})	56
Answer	(ID_M, H, S_{P_A})	88
Answer Forward	(A, ID_C, S_{P_Q})	152

B. Storage

We assume both the baseline and CLARINET log all messages. We additionally assume that the CLARINET implementation uses a queriable log to answer queries. Queriable logs do entail their own overhead, but are becoming more prevalent and vary by implementation, so we discuss only the CLARINET-specific overhead. We compile these components and their scale factor in table XI. N_M is the number of messages sent and received, N_{WM} the number of messages witnessed, N_P the number of known peers, and N_C the number of connections.

1) Messages: Since participants log all messages they send or receive, much of the storage overhead is the same as the network load overhead VIII-A2. In addition to this, participants need to store assessment records to ensure that the harshest reputation operation is always maintained. Because participants store the messages anyway, they can include the assessments as additional information on the message record. An assessment has effectively 4 states, none and the 3 reputation operations, which can be represented as 2 bits, and a message always involves 3 participants so assessments can be stored in a single additional 8-bit integer field[2].

2) Additional: In addition to the assessment records, participants need to maintain records of connections and might keep an explicit reputation score to reduce calculations at the cost of some additional storage. Connection information needs to be retained so participants know the correct peers to reward or penalize and consists of the ID_C, sender, witness, and receiver for a total of $112N_C$ bytes. The reputation can be kept as a double precision floating point with one entry per known peer for a total of $8N_P$ bytes.

3) Witness: While the previous discussion hold for senders and receivers, the baseline system does not involve witnesses, so all messages a participant witnesses entail storage overhead both for data delivery and auditing. Fortunately, it is only the witness's role to verify the claims of the sender or receiver when a dispute should arise. This means that witnesses do not have to store the entire message. Instead, they can store the ID_M, S_S, and a hash of the message, identical to the one they would need anyway for answering queries. This gives an additional storage overhead of $88N_{WM}$ bytes.

4) Total: The total storage for the baseline implementation is $N_M(D_{avg} + S_S)$ bytes. The total supplemental storage for CLARINET is $1953N_M + 112N_C + 8N_P + 2041N_{WM}$ bytes.

$N_C \leq N_M + N_{WM}$ almost certainly holds. The only way it would not is if a participant opens many connections without sending anything on them, and even in that case, peers can remove the connection records after they close because it has no messages to assess and therefore no need to remember the particular participants.

$N_P \leq N_M + N_{WM}$ can hold relatively easily. Participants can use a dynamic default reputation score for peers they have not interacted with in any fashion and participants need at least one message to assess.

Finally, because N_M is the same as in the baseline, the bytes from N_M are fixed overhead relative to the baseline.

This means that the primary source of storage overhead is witnessed messages and grows linearly. While this does somewhat disincentivize being a witness, it is offset by the greater visibility witnesses have into peers' behavior.

IX. EMPIRICAL EVALUATION

In addition to theoretical analysis of a single message, it is important that CLARINET provide tangible benefit at scale. We use PeerSim [29] to implement messaging and reputation largely as discussed. We discuss unspecified behaviors, configuration, and participant actions in the following subsections.

A. Reputation

1) Cooperative: Reputation is implemented as a ratio of two counts, good and total, i.e., $good/total$. All participants begin with 1 in each. Rewards increment both by 1, weak penalties increment total by 1, and strong penalties increment total by 3.

[2]While only 6 bits are completely necessary, many systems do not have much support for integer values less than 8 bits and the 2 additional bits are likely negligible.

TABLE XI
STORAGE SIZES

Component	Implementation	Individual Overhead (bytes)	Scale Factor
Sent & Received Messages	Queriable log of table X	1952	N_M
Witnessed Messages	Queriable log of table X with hash of data	2040	N_{WM}
Assessments	8-bit integer attached to messages	1	$N_M + N_{WM}$
Reputation Scores	Double precisioin floating point	8	N_P
Connection Information	Queriable log	112	N_C

Starting with $1/1$ makes unassessed peers not a special case, defaults to trusting, and allows quick differentiation between weak and strong penalties. Starting at 0, weak $(0/1)$ and strong $(0/3)$ both appear as 0 while 1 gives $1/2$ and $1/4$.

2) Malicious: As discussed in VII-C, we assume all malicious participants are colluding and aware of each other. Because malicious participants do not need to determine non-colluding malicious peers, they do not need to track reputation. We still have them receive forwarded messages and initiate queries, but they take no reputation action based on these.

B. Witness Agreement

1) Cooperative: We take a basic approach to connection setup and witness agreement. Senders always initiate and select the witness. They calculate reputations for all known peers, find the average μ and standard deviation σ of peers with at least one assessment, and then remove peers where $reputation \leq 0.5 \vee reputation < (\mu - \sigma)$. The sender randomly requests witnesses from this trust list until one agrees. Peers only refuse if they already have the maximum number of permitted open connections, which we set to an arbitrary value of 10. If none are able to witness, the sender terminates the connection.

2) Malicious: Malicious participants differ only in witness selection. First they request all colluding peers, simulating influencing witness agreement in favor of a colluder. If none can witness, they randomly select cooperative peers.

C. Configuration

Table XII displays the configurable parameters with values used. We permuted over all possible combinations.

TABLE XII
SIMULATION PARAMETERS

Parameter	Description	Values
Node count	The number of participants in the network	100, 250, 500, 750, 1K, 5K, 10K
Malicious %	Percentage of participants that are malicious	10%, 20%, 30%, 50%, 90%
Malicious action threshold %	Before threshold malicious participants always behave	10%, 20%, 30%, 50%, 70%, 90%
Malicious action %	The probabilistic odds of a malicious participant acting maliciously after the threshold	10%, 20%, 30%, 50%, 70%, 90%

We briefly explain *malicious action threshold %*. This simulates malicious participants that build up reputation prior to misbehaving. We use a simple counter that checks if it is past the threshold. The threshold is the number of rounds the simulation runs multiplied by the percentage.

The counter increments whenever a malicious participant performs an action that may be malicious: sending, witnessing, and responding to a query.

D. Participant Behavior

1) Cooperative: Participants randomly choose between one of the following five actions with equal probability:

- OPEN a connection with a randomly selected peer.
- SEND on a randomly selected open outgoing connection.
- QUERY one peer for a randomly selected message.
- CLOSE a randomly selected outgoing connection.
- Take no action.

If a participant selects an action it cannot successfully complete it takes no action.

2) Malicious: Malicious participants' only difference is they probabilistically act maliciously based on the configurable parameters. This includes sending with an invalid signature, altering witnessed data (including setting sender signature to invalid), and forging query answers. When S and R are malicious, they do not act maliciously because colluders have no reason to deceive each other.

Malicious participants do not consider other actions when deciding to take malicious action. For example, they may supply a valid signature during data delivery and later attempt to claim falsified data. We feel this is a reasonable simplification since messages are not double counted and the simulation focuses on reputation trends. For example a sender that later has a penalized query would in reality have taken malicious action during data delivery, but both cases amount to a single penalty for that message. While this dumb acting would increase the percentage of messages for which a participant behaves maliciously, it is not as dire as it may seem. First, the simulation goal is to observe general trends at scale rather than precise values. Second, it is somewhat offset by the counter potentially not incrementing for a given round. Third, it is additionally offset by the fact that only two of the five actions a participant can take provide any reputation insight.

E. Evaluation Results

We collected the following data that captures participants' views of the network:

- Whether the participant was cooperative or malicious

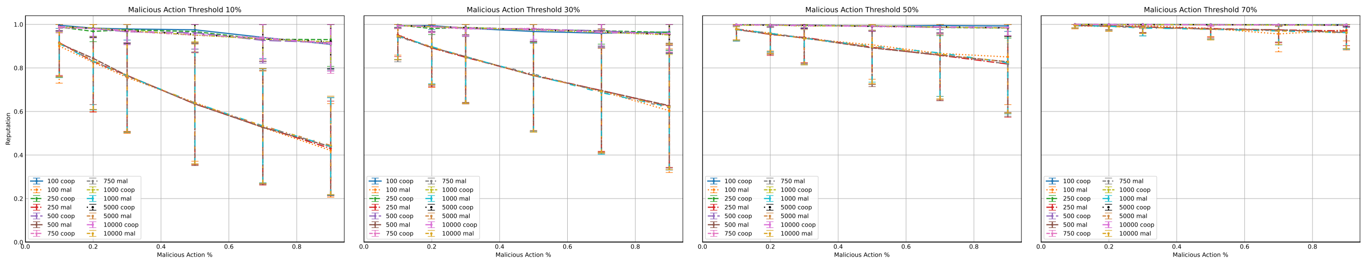

Fig. 2. Reputation as affected by how long malicious participants wait to misbehave.

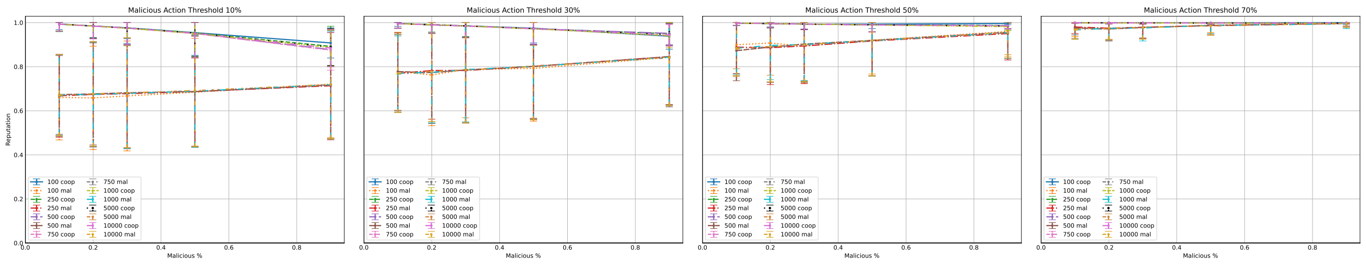

Fig. 3. Reputation as affected by percentage of the network that is malicious.

- Individual peer information such as:
  - The peer's reputation with the participant
  - Whether the peer was cooperative or malicious
- Aggregate information for assessed peers:
  - Average reputation
  - Minimum reputation
  - Maximum reputation
  - Standard deviation

One set of aggregate information exists for each of all, cooperative, and malicious assessed peers. We used PeerSim's global knowledge to facilitate this, but cooperative participants were not aware of peer status during the simulation.

We consolidated this data by averaging the aggregate statistics of all the cooperative participants within a given simulation for their cooperative and malicious peers. We then averaged this over all simulations with the same number of participants.

Figures 2 and 3 plot reputation against another parameter. Each separates the simulations by the malicious action threshold percentage. We omit 20% simply to reduce the number of figures and 90% because it did not display meaningful results. It appeared that 90% was so high that nearly no malicious participants misbehaved, resulting in nearly all peers having the max reputation of 1.0.

Both show that as malicious participants wait longer to begin acting maliciously it becomes harder to detect them. This is expected because waiting longer means there are fewer malicious actions to penalize. Even 70%, despite seeming quite similar, is still encouraging. Because only two out of the five actions a participant or its peers could take contribute to reaching this threshold, malicious participants do not necessarily begin acting maliciously 70% of the way through the simulation. Some may reach the threshold earlier and others later. Given the odds of taking an action that does not contribute, we suspect that malicious participants more commonly reached the threshold over 70% of the way through the simulation.

The figures also show that CLARINET is stable with regards to network size over a given time frame. This is mainly due to the measurements being based on interaction with peers. Interaction occurs either when the participant reaches out to a peer or when a peer contacts the participant. In a set amount of time, a participant is likely to reach out to the same number of peers regardless of network size. While a larger network means more peers could potentially contact the participant, these peers have more other peers they could also potentially contact, offsetting the increased odds of any peer contacting a particular participant.

Figure 2 demonstrates that as peers act maliciously more often their reputation decreases more significantly. Cooperative peers suffer some decrease as well because of erroneous weak penalties, but are significantly more stable at a higher overall reputation, indicated by a smaller standard deviation. Malicious peers likely have a larger standard deviation due to the variance in whether they acted maliciously and whether the cooperative participants happened to assess them.

Figure 3 shows that as the percentage of malicious participants increases their overall reputation also increases while cooperative peers' reputations decrease. Malicious peers' reputation may increase because they do not act maliciously when S and R are both malicious. Because more peers are malicious, it increases the odds of this, decreasing the overall malicious activity that cooperative participants witnessed. Additionally, because there are more malicious participants, penalties may be spread over a larger number of peers resulting in fewer per malicious peer. Similarly, because there are fewer cooperative participants, erroneous false penalties may be more concentrated resulting in more per peer. Even still, there is a noticeable gap between cooperative and malicious peers.

X. Limitations and Future Work

ClariNet is entirely reactive so can only guard against repeat offenders. In the worst case, the witness is a colluder, and the participant has no recourse should a dispute arise. However, protection from repeat offenders is better than no protection and participants' ability to gain insight as a witness helps them assess peers without being at risk.

Ideally, ClariNet would support multiple witnesses, but we limited the analysis to one to constrain complexity. We believe this is reasonable for an initial exploration. Supporting multiple witnesses is a prime area for future work.

The current analysis does not investigate malicious participants blocking communication. While ClariNet does have some protections like fully encrypted messages, heuristics can still be effective. Future work could investigate potential heuristics, their effectiveness, and potential mitigations.

The analysis also does not factor in malicious participants that claim unsent messages. Participants cannot assess messages they are unaware of. While they may be able to take certain steps like using received queries as query candidates, this requires additional analysis to ensure it is not exploitable and would still not guard against entirely fabricated messages.

ClariNet currently requires a reasonably stable network. In highly dynamic networks, participants may not have enough time to conduct extensive auditing to form solid peer reputations. The benefits from serving as a witness do encourage participants to stay active, but this is not always practical.

Participants could likely make more intelligent decisions based on combinations of messages to prevent erroneous penalties or better apply strong penalties. While limiting these conditions for initial analysis was reasonable to reduce complexity, investigating and analyzing more intelligent conditions would be an area for future work.

The empirical analysis was limited. While randomized traffic can approximate large-scale systems, drawing from real-world dynamic distributed systems would provide better data. We intend to investigate the system with more intelligent malicious participants and build a real, deployable version of ClariNet to test in a real distributed environment.

XI. Related Work

While all the building blocks ClariNet uses have been extensively studied and even combined, to our knowledge no existing works combine them in an eventual-consistency model geared toward audit logging the way ClariNet does. The majority of similar works are orthogonal and can potentially be combined with ClariNet to gain the benefits of both.

Several works [1]–[4] address non-repudiation by ensuring that neither side is able to gain an advantage during the data exchange. They often do this by conducting iterative, bit-by-bit transfer so the message either side wishes to receive is not usable until it is completely received. This ensures that either both sides obtain proof of exchange or that neither side has anything. ClariNet allows senders to lazily obtain proof of receipt when they have capacity and allows for full transmission of data at once.

Optimistic fair exchange [21] provides remuneration should one party attempt to cheat through the presence of an arbitrator, but is reliant on a trustworthy arbitrator. Some works [22] address malicious arbitrators by ensuring the arbitrator can be held accountable. ClariNet gives participants insight into peer trustworthiness when making an initial selection.

More recent work on non-repudiation [27], consensus protocols, and consensus reputation [37]–[39] and surveys [14]–[16] leverage blockchain [20]. While robust, this approach has raised environmental concerns [26] and the overhead can introduce noticeable latency [24], [25]. As such, blockchain may not be desirable. ClariNet does not seek to supplant such systems; rather it can provide a less resource-intensive alternative.

Non-blockchain consensus and Byzantine agreement protocols [17]–[19] are typically older works often designed with different goals, such as accommodating faulty hardware rather than active manipulation or upper-bounding agreement time. ClariNet detects malicious parties so they can be excluded from future committee construction.

There is a wealth of work on reputation. Some focuses on narrow domains [13], but much is generalized peer-to-peer networks. Some address areas like balancing reputation and privacy [23]. Others on finding reliable resource providers [5]–[7]. Others aim to represent some generalized reputation score separate from specific systems [8]–[12]. All focus on securely sharing reputations rather than calculating initial reputations. ClariNet aims to provide a specific, generalizable, and tamper-resistant calculation algorithm. In this regard, it is similar to Guru [40], but Guru requires consensus rounds while ClariNet allows for eventual consistency and stronger insight at the cost of a more limited consensus committee.

XII. Conclusion

In this paper, we presented ClariNet, a novel reputation scheme that is highly general and can identify malicious peers even in the face of uncertainty. ClariNet aims to provide defense in depth by layering common security approaches such as cryptographic signatures, authenticated key exchange, a minimal consensus protocol, a reputation scheme, and log auditing to help participants defend against false accusations by malicious peers. ClariNet allows participants to, on average, differentiate malicious and cooperative peers even when malicious peers misbehave infrequently or even dominate the network. Additionally, ClariNet does not place strict deadlines on many of its operations. This allows participants to separate data delivery from auditing, reducing latency in processing, and perform auditing at their discretion, such as deferring it until times when load is low. While ClariNet may not be universally applicable, it can assist participants in distributed systems where peers are not implicitly trusted.

References

[1] M. Ben-Or, O. Goldreich, S. Micali, and R. L. Rivest, "A fair protocol for signing contracts," IEEE Transactions on Information Theory, vol. 36, no. 1, pp. 40–46, Jan. 1990, doi: 10.1109/18.50372.

[2] O. Markowitch and Y. Roggeman, “Probabilistic Non-Repudiation without Trusted Third Party”.

[3] T. Coffey and P. Saidha, “Non-repudiation with mandatory proof of receipt,” SIGCOMM Comput. Commun. Rev., vol. 26, no. 1, pp. 6–17, Jan. 1996, doi: 10.1145/232335.232338

[4] S. Kremer, O. Markowitch, and J. Zhou, “An intensive survey of fair non-repudiation protocols,” Computer Communications, vol. 25, no. 17, pp. 1606–1621, Nov. 2002, doi: 10.1016/S0140-3664(02)00049-X.

[5] M. Gupta, P. Judge, and M. Ammar, “A reputation system for peer-to-peer networks,” in Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video, in NOSSDAV ’03. New York, NY, USA: Association for Computing Machinery, Jun. 2003, pp. 144–152. doi: 10.1145/776322.776346.

[6] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, “A reputation-based approach for choosing reliable resources in peer-to-peer networks,” in Proceedings of the 9th ACM conference on Computer and communications security, in CCS ’02. New York, NY, USA: Association for Computing Machinery, Nov. 2002, pp. 207–216. doi: 10.1145/586110.586138.

[7] F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, “Choosing reputable servents in a P2P network,” in Proceedings of the 11th international conference on World Wide Web, in WWW ’02. New York, NY, USA: Association for Computing Machinery, May 2002, pp. 376–386. doi: 10.1145/511446.511496.

[8] A. A. Selcuk, E. Uzun, and M. R. Pariente, “A reputation-based trust management system for P2P networks,” in IEEE International Symposium on Cluster Computing and the Grid, 2004. CCGrid 2004., Apr. 2004, pp. 251–258. doi: 10.1109/CCGrid.2004.1336575.

[9] K. Aberer and Z. Despotovic, “Managing trust in a peer-2-peer information system,” in Proceedings of the tenth international conference on Information and knowledge management, in CIKM ’01. New York, NY, USA: Association for Computing Machinery, Oct. 2001, pp. 310–317. doi: 10.1145/502585.502638.

[10] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, “The Eigentrust algorithm for reputation management in P2P networks,” in Proceedings of the 12th international conference on World Wide Web, in WWW ’03. New York, NY, USA: Association for Computing Machinery, May 2003, pp. 640–651. doi: 10.1145/775152.775242.

[11] R. Zhou, K. Hwang, and M. Cai, “GossipTrust for Fast Reputation Aggregation in Peer-to-Peer Networks,” IEEE Transactions on Knowledge and Data Engineering, vol. 20, no. 9, pp. 1282–1295, Sep. 2008, doi: 10.1109/TKDE.2008.48.

[12] T. Beth, M. Borcherding, and B. Klein, “Valuation of trust in open networks,” in Computer Security — ESORICS 94, D. Gollmann, Ed., Berlin, Heidelberg: Springer, 1994, pp. 1–18. doi: 10.1007/3-540-58618-0_53.

[13] L. Xiong and L. Liu, “A reputation-based trust model for peer-to-peer ecommerce communities [Extended Abstract],” in Proceedings of the 4th ACM conference on Electronic commerce, in EC ’03. New York, NY, USA: Association for Computing Machinery, Jun. 2003, pp. 228–229. doi: 10.1145/779928.779972.

[14] A. Singh, G. Kumar, R. Saha, M. Conti, M. Alazab, and R. Thomas, “A survey and taxonomy of consensus protocols for blockchains,” Journal of Systems Architecture, vol. 127, p. 102503, Jun. 2022, doi: 10.1016/j.sysarc.2022.102503.

[15] J. Xu, C. Wang, and X. Jia, “A Survey of Blockchain Consensus Protocols,” ACM Comput. Surv., vol. 55, no. 13s, p. 278:1-278:35, Jul. 2023, doi: 10.1145/3579845.

[16] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, “A Survey of Distributed Consensus Protocols for Blockchain Networks,” IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 1432–1465, 2020, doi: 10.1109/COMST.2020.2969706.

[17] D. Dolev and H. R. Strong, “Authenticated Algorithms for Byzantine Agreement,” SIAM J. Comput., vol. 12, no. 4, pp. 656–666, Nov. 1983, doi: 10.1137/0212045.

[18] P. Feldman and S. Micali, “Optimal algorithms for Byzantine agreement,” in Proceedings of the twentieth annual ACM symposium on Theory of computing, in STOC ’88. New York, NY, USA: Association for Computing Machinery, Jan. 1988, pp. 148–161. doi: 10.1145/62212.62225.

[19] G. Bracha and S. Toueg, “Resilient consensus protocols,” in Proceedings of the second annual ACM symposium on Principles of distributed computing, in PODC ’83. New York, NY, USA: Association for Computing Machinery, Aug. 1983, pp. 12–26. doi: 10.1145/800221.806706.

[20] “Bitcoin Whitepaper.” Accessed: Oct. 03, 2024. [Online]. Available: https://portfolio-blog-starter.vercel.app/blog/bitcoin-whitepaper

[21] N. Asokan, V. Shoup, and M. Waidner, “Optimistic fair exchange of digital signatures”.

[22] X. Huang, Y. Mu, W. Susilo, W. Wu, J. Zhou, and R. H. Deng, “Preserving Transparency and Accountability in Optimistic Fair Exchange of Digital Signatures,” IEEE Transactions on Information Forensics and Security, vol. 6, no. 2, pp. 498–512, Jun. 2011, doi: 10.1109/TIFS.2011.2109952.

[23] L. Lu et al., “Pseudo Trust: Zero-Knowledge Authentication in Anonymous P2Ps,” IEEE Transactions on Parallel and Distributed Systems, vol. 19, no. 10, pp. 1325–1337, Oct. 2008, doi: 10.1109/TPDS.2008.15.

[24] J. Chen and Y. Qin, “Reducing Block Propagation Delay in Blockchain Networks via Guarantee Verification,” in 2021 IEEE 29th International Conference on Network Protocols (ICNP), Nov. 2021, pp. 1–6. doi: 10.1109/ICNP52444.2021.9651926.

[25] X. Zhang et al., “The Block Propagation in Blockchain-Based Vehicular Networks,” IEEE Internet of Things Journal, vol. 9, no. 11, pp. 8001–8011, Jun. 2022, doi: 10.1109/JIOT.2021.3074924.

[26] M. Wendl, M. H. Doan, and R. Sassen, “The environmental impact of cryptocurrencies using proof of work and proof of stake consensus algorithms: A systematic review,” Journal of Environmental Management, vol. 326, p. 116530, Jan. 2023, doi: 10.1016/j.jenvman.2022.116530.

[27] F. Chen, J. Wang, J. Li, Y. Xu, C. Zhang, and T. Xiang, “TrustBuilder: A non-repudiation scheme for IoT cloud applications,” Computers & Security, vol. 116, p. 102664, May 2022, doi: 10.1016/j.cose.2022.102664.

[28] C. Nist, “The digital signature standard,” Commun. ACM, vol. 35, no. 7, pp. 36–40, Jul. 1992, doi: 10.1145/129902.129904.

[29] A. Montresor and M. Jelasity, “PeerSim: A Scalable P2P Simulator,” in Proc. of the 9th Int. Conference on Peer-to-Peer (P2P’09), Seattle, WA, Sep. 2009, pp. 99–100.

[30] “Peers,” libp2p. Accessed: Nov. 14, 2024. [Online]. Available: https://docs.libp2p.io/concepts/fundamentals/peers/

[31] W. Diffie, P. C. Van Oorschot, and M. J. Wiener, “Authentication and authenticated key exchanges,” Des Codes Crypt, vol. 2, no. 2, pp. 107–125, Jun. 1992, doi: 10.1007/BF00124891.

[32] “The Noise Protocol Framework.” Accessed: Nov. 14, 2024. [Online]. Available: https://noiseprotocol.org/noise.html

[33] T. Perrin, “The Noise Protocol Framework”.

[34] “An open system to manage data without a central server,” IPFS. Accessed: Nov. 14, 2024. [Online]. Available: https://ipfs.tech

[35] J. Iyengar and M. Thomson, “QUIC: A UDP-Based Multiplexed and Secure Transport,” Internet Engineering Task Force, Request for Comments RFC 9000, May 2021. doi: 10.17487/RFC9000.

[36] “SPDY: An experimental protocol for a faster web.” Accessed: Nov. 19, 2024. [Online]. Available: https://www.chromium.org/spdy/spdy-whitepaper/

[37] W. Cai, W. Jiang, K. Xie, Y. Zhu, Y. Liu, and T. Shen, “Dynamic reputation–based consensus mechanism: Real-time transactions for energy blockchain,” International Journal of Distributed Sensor Networks, vol. 16, no. 3, p. 1550147720907335, Mar. 2020, doi: 10.1177/1550147720907335.

[38] [X. Wang and Y. Guan, “A Hierarchy Byzantine Fault Tolerance Consensus Protocol Based on Node Reputation,” Sensors, vol. 22, no. 15, Art. no. 15, Jan. 2022, doi: 10.3390/s22155887.

[39] Q. Zhuang, Y. Liu, L. Chen, and Z. Ai, “Proof of Reputation: A Reputation-based Consensus Protocol for Blockchain Based Systems,” in Proceedings of the 1st International Electronics Communication Conference, in IECC ’19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 131–138. doi: 10.1145/3343147.3343169.

[40] A. Biryukov, D. Feher, and D. Khovratovich, “Guru: Universal Reputation Module for Distributed Consensus Protocols”.