BAR Gossip

Harry C. Li, Allen Clement, Edmund L. Wong, Jeff Napper, Indrajit Roy,
Lorenzo Alvisi, Michael Dahlin
Dept. of Computer Sciences, The University of Texas at Austin

We present the first peer-to-peer data streaming application that guarantees predictable throughput and low latency in the BAR model, in which non-altruistic nodes can behave in a self-serving (rational) or even arbitrarily malicious (Byzantine) way. At the core of our solution is a BAR-tolerant version of gossip, a well-known technique for scalable and reliable data dissemination. In traditional gossip, data dissemination is performed with randomly selected partners; such non-determinism, we show, offers an excellent opening to rational nodes bent on gaming the system. BAR-tolerant gossip instead relies on a verifiable pseudo-random partner selection mechanism that eliminates non-determinism while maintaining the unpredictability and rapid convergence of traditional gossip. Our initial experience indicates that BAR Gossip is robust against up to 20% of nodes exhibiting Byzantine behavior and even against up to 40% of nodes colluding together against the remaining nodes. In either case, our BAR-tolerant video streaming application provides over 95% convergence for broadcast updates.

1 Introduction

Streaming media is an increasingly useful application at several scales of deployment. For example, the 2006 NCAA tournament had peak participation of 150k+ users for their live streaming services. At smaller scales, such as academic conferences like OSDI or artistic events like Austin's SXSW, venues draw audiences of dozens to hundreds of people.

At all these scales, a peer-to-peer (p2p) streaming solution appears to be an intriguing alternative to traditional methods. One advantage of p2p systems is their potential to be highly robust, scalable, and adaptive. For example, a p2p architecture could, in principle, absorb the impact of an unexpected flash crowd. Furthermore, large-scale content providers may adopt p2p-based solutions to shift costs (like bandwidth) to clients, and small-scale providers might find it simpler to use a self-organizing p2p network instead of maintaining a dedicated server.

Realizing the promises of p2p in streaming services is a non-trivial task. First, the service needs to guarantee highly reliable, stable, and timely throughput of messages despite the presence of faulty, misconfigured, or even malicious peers. Second, the service must be robust against selfish users, who try to catch a *free ride* by receiving streams without contributing their fair share to others [16]. The freerider phenomenon in p2p systems is indeed a symptom of a broader issue: any system in which nodes are not under the control of a single administrative domain must be designed for the possibility that nodes will deviate from their specification if doing so is to their advantage. File-sharing p2p applications, like BitTorrent [10], have recognized this issue and introduced a set of heuristics to induce rational nodes to comply with the protocol—these heuristics, however, optimize for large file transfer, not the timely delivery of a series of relatively small frames required by a streaming service.

This paper presents the first p2p streaming media application designed for a system model (BAR [1]) in which altruistic nodes (who follow the protocol assigned to them) coexist with both Byzantine and rational nodes. Our BAR-tolerant solution is based on gossip protocols [5, 12, 25, 41]. The protocol is simple to understand and implement, and provides a scalable mechanism for information dissemination that ensures predictable throughput—even if most or all of the nodes act selfishly and the remainder act maliciously or malfunction in arbitrary ways.

The defining characteristic of gossip protocols is that each node exchanges data, or gossips, with randomly selected peers: it is precisely this randomness that gives gossip protocols their enviable robustness. From the perspective of designing BAR-tolerant protocols, however, randomness can be a real headache: in fact, *any* source of non-determinism is hard to deal with in the BAR model because it gives opportunities for rational users to hide selfish actions in the guise of legitimate, non-deterministic behavior.

We overcome this difficulty by building a BAR-tolerant gossip protocol that uses *verifiable pseudo-randomness* as the means for peer selection: in particular, we exploit properties of pseudo-random number generators and unique signature schemes to build a verifiable pseudo-random partner selection algorithm. This approach eliminates the main source of non-determinism in traditional gossip—randomness in partner selection—yet maintains the unpredictability and rapid convergence of traditional gossip. Our novel peer selection technique works in combination with a simple mechanism designed to encourage the fair exchange [22] of one node's updates for another's—its simplicity is achieved by doing away with reputations [21, 29, 42] and by leveraging instead the notion of credible threats [1, 13].

We build a prototype streaming application that uses our BAR Gossip protocol to provide a stable throughput multicast. We show that BAR gossip is robust to Byzantine and selfish behavior, even when a significant number of selfish nodes collude. Additionally, we demonstrate that our protocol can reduce the bandwidth requirements of a provider of live streaming content by 95% in an environment where 20% of nodes are Byzantine and the remaining nodes are rational, while incurring only a 10 second end-to-end latency.

This paper makes the following contributions:

- We present the first p2p streaming application in the BAR model.
- We design the first gossip protocol in the BAR model.

We introduce this paper in Section 1, which ends with this paragraph outlining the rest of the paper. Section 2 frames our contributions in the context of previous works. We define the system model in Section 3 . Section 4 describes the BAR Gossip protocol. Section 5 supports our claim that BAR Gossip is a robust streaming protocol through a combination of simulations and live experiments.

2 Related Work

BAR Gossip targets streaming of live content among Byzantine, altruistic, and rational nodes. It draws on a broad literature of bulk file transfer systems designed to tolerate node misbehavior as well as a large number of efforts to use gossip for robust data dissemination.

Bulk file transfer. Several systems have addressed selfish behavior in p2p content distribution. BitTorrent [9] implements a tit-for-tat incentive structure in which nodes, when sharing the content of a particular file, give preference to nodes who have reliably reciprocated in the past. Scrivener [34] generalizes BitTorrent by supporting a distributed reputation scheme based on credits that can be earned and redeemed across multiple files: through this mechanism, a Scrivener node that has been a good citizen can enlist the help of its peers even if the file it wants to acquire is not popular.

FOX [24] guarantees optimal download time to all the nodes interested in acquiring the same popular file under the assumption that all nodes are selfish. This impressive guarantee, however, is achieved at the expense of robustness: the system's incentive structure depends on the fear of mutual assured destruction, and a single Byzantine or simply malfunctioning node can cause the entire system to collapse.

Splitstream [7] is a tree-based multicast protocol that achieves load balancing by dividing content into multiple stripes, each of which is multicast using a separate tree. Splitstream is vulnerable to freeloaders. Ngan et al. [37] observe that if Splitstream's multicast trees are periodically rebuilt and nodes remember the upstream nodes that misbehaved, upstream nodes in a given tree have an incentive to provide good service to nodes downstream to avoid future retaliation.

Habib and Chuang [20] study p2p streaming of non-live media. They assume a set of supplier nodes that know the entire stream: their goal is to help a receiver select, among the suppliers, those most likely to provide a high quality stream.

BAR Gossip differs from these systems in three key ways. First, these systems work to optimize average download bandwidth over long periods of time and do not attempt to maintain stable throughput over shorter intervals. In contrast, our protocol is designed to disseminate live streams and therefore values the highly reliable, stable, and timely throughput that comes with gossip-style data dissemination. Second, several of these systems are designed to be robust to Byzantine [7] or rational [20, 24, 34] players but not both. Third, all of these systems transfer a large collection of file blocks. In contrast, BAR Gossip distributes live streams and must cope with having a relatively small window of "useful" data in flight at any given time; ensuring timely delivery of a small set of data is one of the key challenges in our protocol. Fourth, most of these systems make use of local [9, 37] or distributed [20, 34] node reputations in their incentive structure. But given our desire to provide stable throughput over short periods of time, it is problematic for us to rely on a node's long-term reputation to predict its short-term behavior. Furthermore, because gossip partners are likely to change in every round, in our protocol it is virtually impossible for a node to build enough good will with specific partners to support a purely local reputation scheme à la BitTorrent. We concern ourselves with both rational and Byzantine players—an environment where designing and implementing a strategy-proof reputation system appears complex.

Gossip. Gossip algorithms were first introduced by Demers et al. [12] to manage replica consistency in the Xerox Clearinghouse Service [38]. Following Birman et al.'s highly influential paper on Bimodal Multicast [5] gossip algorithms have established themselves as one of the leading approaches to achieve reliable and scalable application-level multicast [3, 6, 8, 15, 18, 25, 31, 43, 46]. Gossip protocols are also at the core of a new generation of scalable distributed protocols for failure detection [41], group communication [17, 45], and the monitoring and management of large distributed systems [44].

Experience with the CoolStream implementation of the DONet p2p overlay network [47] makes a strong case for the scalability of gossip-based dissemination of live data streams and for its potential to deliver high quality end-to-end user experience in the presence of purely altruistic nodes.

Secure gossiping aims at preventing Byzantine nodes from spreading false gossips. While digital signatures are not necessary to accomplish this goal [26–28, 32], they considerably simplify protocol design and are assumed in most practical gossip-based systems [6, 44, 45], including BAR Gossip.

DRUM [2] assumes secure gossip and focuses on fighting denial of service (DoS) attacks through two main techniques: bounding the amount of resources allocated to each gossip operation and directing these operations to random ports. While in this paper we do not explicitly address DoS attacks, we believe that BAR Gossip would be able to leverage ideas from DRUM to reduce its vulnerability to this threat.

BAR replication. Aiyer et al. [1] define a replicated state machine protocol under the BAR model. Although the replication overheads of that approach are too high for our target application, we draw on many of the same design principles that were useful there: *predictable communication patterns* manifests in our partner selection algorithm (Section 4.1.1), *cost balancing* manifests in our optimistic push algorithm (4.1.6), *credible threats* manifest in our key exchange (4.1.4), and *ensuring long term benefit* manifests in our deferred gratification protocol structure (4.1).

3 Model

We consider the problem of streaming a live event across the Internet where, to reduce the source's broadcast costs, the audience helps disseminate the stream by forming a peer-to-peer network. We assume the broadcaster is *altruistic* (i.e., always follows the specified protocol [1]) and multicasts $bcast_{cnt}$ updates every round to a constant fraction $bcast_{hr}$ of the audience, which consists of any fraction of altruistic, rational, and Byzantine nodes. We assume that non-Byzantine nodes are interested in receiving an update only within the first upd_{dl} seconds since the update was multicast. After this time, the non-Byzantine nodes who have received the update desist from trying to disseminate it further and deliver it to their media player; we say that the update has *expired*.

Each *rational* node follows a strategy that maximizes its utility. The utility function describes the costs and benefits and relative weights as viewed by a rational node; in this paper, the benefit consists in the ability to play the live stream and the costs are incurred by sending and receiving packets. Rational nodes participate in the stream dissemination protocol if their end-to-end benefit in playing the stream exceeds the communication costs—in particular, we assume that for rational nodes the benefit received by acquiring each update exceeds the communication costs incurred in doing so.

Altruistic nodes follow the protocol as given regardless of costs, similar to correct nodes in the traditional fault-tolerance literature, while *Byzantine* nodes behave arbitrarily.

Nodes maintain clocks synchronized within δ seconds of each other and communicate over point-to-point, synchronous, and unreliable links using both TCP and UDP. When messages are exchanged using UDP, a node that does not receive an expected message assumes that the link dropped it.

To mitigate the threat of Sybil attacks [14], it must be hard for malicious (or greedy) participants to collect multiple identities. In our prototype, this is accomplished by limiting each IP address to at most one identity; we model scenarios in which a participant is able to obtain multiple IP addresses as an instance of collusion between nodes—we discuss collusion and its effects in Sections 5.5 and 5.6.

We assume that nodes subscribe to the live broadcast prior to its start and that non-Byzantine nodes remain in the system for the duration of the broadcast—in other words, we assume a static membership system.

Before the broadcast begins, the broadcaster assigns public/private keys to each node (including itself) and creates a membership list that contains the identity of each participant node as well as its public key. Each node obtains from the broadcaster the membership list, together with its own private key, prior to the event's start. Nodes sign protocol messages using these asymmetric keys to provide authentication and guarantee message integrity; likewise, the broadcaster signs stream updates to prevent forgeries. We make the standard assumption that the cryptographic primitives used in our protocol (which also include one-way hash functions)

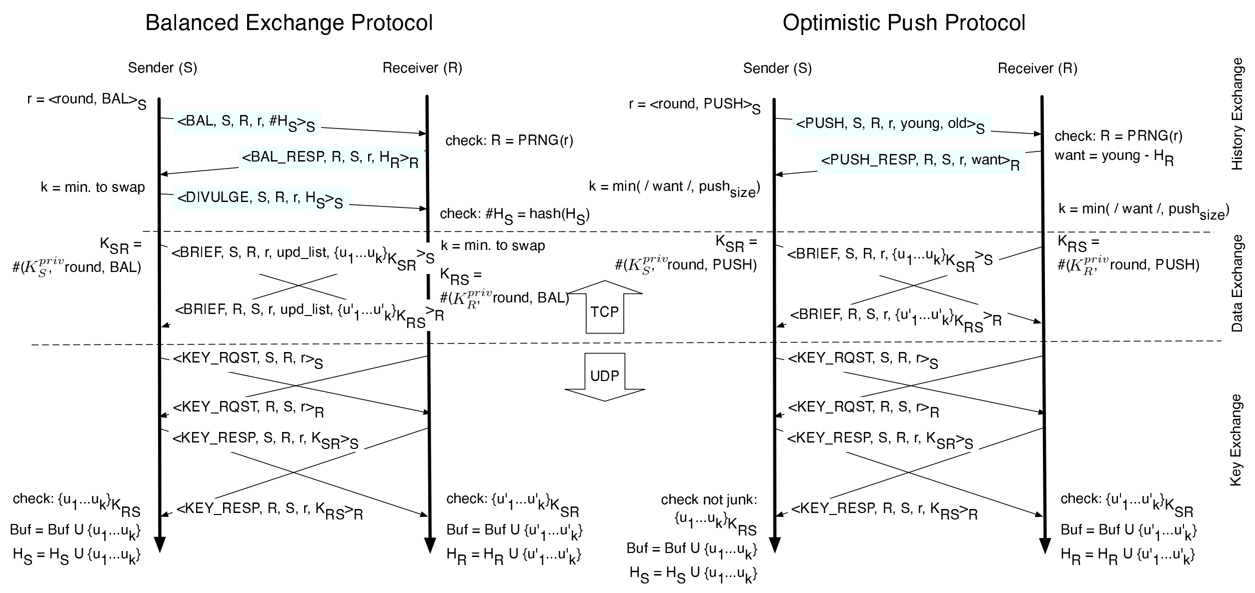

Figure 1: Balanced Exchange and Optimistic Push Protocols for node A contacting node B. We denote hash(M) by #M, encrypt(M) with key K by $\{M\}_K$, and sign(M) by A using $\langle M \rangle_A$.

cannot be subverted.

An important property of our protocol is that it requires signatures to be unique: for a given m, there must exist exactly one valid signature of m by i. Although a number of standard signature algorithms fail to provide this property [36], there exist algorithms that do [4]. We denote a message m signed by i as $\langle m \rangle_i$.

Signed messages that are internally inconsistent with the protocol amount to proofs of misbehavior (POM). For example, a POM could be a signed message claiming the encrypted content of a message is one thing, whereas the actual content is garbage data. The broadcaster *evicts* every node for which the broadcaster has a POM by including in the stream signed eviction notices. Non-Byzantine nodes have no interest in communicating with evicted nodes.

4 BAR Gossip Design

We begin by giving a high-level description of BAR Gossip from the perspective of an altruistic node. In the interest of clarity we ignore, for the moment, that some of the nodes in our system may behave selfishly or arbitrarily—our system may, in fact, include no altruistic nodes! We discuss how the protocol tolerates rational and Byzantine behavior later in this section.

We structure BAR-Gossip as a sequence of $T + \delta$-long rounds, where T is a time interval sufficient to complete the message exchanges required by the "Balanced Exchange Protocol" and "Optimistic Push Protocol" outlined in Figure 1 (in our prototype, each round lasts one second).

At the beginning of round r, each node delivers to its media player the updates that expired in round $r - 1$. A node then concurrently executes three tasks: *i)* waiting for the possible receipt of new updates from the broadcaster; *ii)* participating in the balanced exchange of non-expired updates with other nodes in the audience—each node initiates one such exchange with one randomly selected audience node; *iii)* participating in optimistic pushes of non-expired updates with other nodes in the audience—again, each node initiates one such exchange with one randomly selected audience node.

Figure 1 shows that the two exchange protocols have the same structure. First, the sender S selects a partner (the receiver R); S and R then engage each other in three trades: the *history exchange*, in which the two parties learn about the updates the other party holds; the *update exchange*, in which each party copies a subset of these updates into a *briefcase* that is sent, encrypted, to the other party; and the *key exchange*, where the parties swap the keys needed to access the updates in the briefcases. For both exchange protocols, the keys used to encrypt the briefcases are symmetric and generated by hashing the sender's private key, a unique seed value, and the protocol variation (Balanced vs. Optimistic).

The Balanced Exchange and Optimistic Push Protocols differ, however, in what the parties disclose to each other during the *history exchange* and in how they determine the content of their respective briefcases during the *update exchange*.

In a balanced exchange, each party determines the largest number of new updates it can receive while keeping the exchange fair—each party forwards an update to its partner only if it receives in return an update it had not seen before. Each party therefore sends to the other a *history* set H containing the identifiers of all the updates it currently holds, compares the history it has received with its own, and determines the largest number k of updates that can be exchanged on a one-for-one basis. It then puts in its briefcase the k most recent updates unknown to its partner and encrypts them as described above.

Optimistic Push helps nodes that have fallen behind in the broadcast and that may not have any updates to trade in a Balanced Exchange. We call this exchange protocol Optimistic because in it S, who initiates the protocol, is willing to forward useful updates to R without first ascertaining whether R can return the favor. In particular for the history exchange, S forwards to R two lists: a *young list*, which contains the identifiers of some of the most recent updates S knows, and an *old list*, which contains the identifiers of updates that S is missing and that are about to expire. R replies with a *want list*, which contains the identifiers of the updates in the *young list* that the Receiver is actually missing. S and R then exchange briefcases: the sender's contains the k updates in the *want list*. Optimistic Push leaves more flexibility to the receiver in determining the content of the briefcase it sends: a receiver who has fewer than k of the updates listed in the sender's *old list* has the option of sending back up to an equivalent number of *junk* updates. It is this flexibility that allows a receiver that has fallen behind and has nothing to trade in balanced exchanges to catch up.

We regulate Optimistic Push with two parameters, *push*$_{age}$ and *push*$_{size}$: the *young list* consists only of updates that have been broadcast within the last *push*$_{age}$ rounds and *push*$_{size}$ is an upper limit on the number of updates that the Receiver can place in its *want list*. Larger values of *push*$_{size}$ help lagging nodes to catch up faster; however, they also increase the likelihood that such nodes will waste bandwidth by sending junk and make the protocol more vulnerable to denial of service attacks launched by Byzantine nodes.

We now explain how BAR Gossip addresses both rational and Byzantine behaviors. We discuss robustness against rational behavior within the framework of Nash equilibria. In a Nash equilibrium [35], no node has an incentive to *unilaterally* deviate from the equilibrium, assuming every other node follows the protocol. One weakness of this approach is that it ignores the possibility that a node may have such an incentive if it can convince some other node to also deviate with it. We investigate the effects of such collusions in Section 5.

4.1 Designing for Rational Behavior

Our approach to making BAR Gossip robust to rational behavior follows two principles. First, we *restrict choice* within balanced exchanges and optimistic pushes. Restricted choice provides safety properties: if a rational node decides to participate in an exchange, then it sends only messages as prescribed by the protocol. Second, because a rational node only participates in protocol steps to obtain benefit, we *delay gratification* in both protocols until the key exchange. Delayed gratification provides the liveness properties: a rational node that chooses to participate in an exchange will participate in all steps of the exchange because doing so eventually yields a net benefit.

The design of the Balanced Exchange protocol exemplifies these principles. As discussed in the previous section, the protocol consists of four phases: partner selection, history exchange, update exchange, and key exchange. The first three of these phases use TCP as the underlying send/receive primitive. In contrast, the fourth phase uses UDP. We explain this choice when we discuss the key exchange.

It is easy to see that the Balanced Exchange protocol uses delayed gratification: a node only gains access to the updates contained in its partner's briefcase during the last phase of the protocol; furthermore, a node that goes faithfully through the protocol's four phases does indeed contribute to the dissemination of updates.

In the following subsections, we examine each phase of the Balanced Exchange Protocol in detail. We give the intuition behind our design choices and formally state the properties of each phase at the end of each subsection. Together, the lemmas show that a rational node participating in a balanced exchange faithfully follows each step.

Then, in Section 4.1.6, we briefly reprise our analysis for the Optimistic Push Protocol. Although restricted choice still limits the messages that a rational node will send, the extra flexibility of the Optimitic Push Protocol makes faithful participation less certain. Note that although our analysis cannot prove that initiating and faithfully responding to optimistic pushes is the rational choice, our experimental evidence in Section 5 strongly suggests it is.

4.1.1 Partner Selection

Problem: *What if a rational node selects more partners per round than prescribed or biases its selections instead of choosing partners uniformly at random?*

Partner selection highlights a fundamental difference between traditional gossip and BAR Gossip. In a traditional gossip protocol, each node periodically selects a partner using a pseudo-random number generator

(PNRG) and contacts that partner to request an exchange of information. Each node also generally accepts every gossip request it receives up to some per round maximum.

Random partner selection lies at the core of gossip's robustness—it is what yields its rapid convergence, fault-tolerance, and stable throughput. However, when we consider rational behavior, this strength can become a weakness. Rational nodes may have little incentive to adhere to a traditional partner selection algorithm, thereby dissolving gossip's robustness guarantees.

We address this challenge by focusing on the properties of the value used to seed the PRNG. Traditonally, gossip protocols use non-deterministic seed values to achieve unpredictability, thereby limiting the effectiveness of targeted attacks. Non-determinism, however, leaves an opening for rational nodes to game the system. BAR Gossip manages to keep the value unpredictable while making it deterministic and therefore verifiable.

We now look in detail at the Balanced Exchange Protocol (similar arguments hold for the Optimistic Push Protocol). In BAR Gossip, the sender S selects its balanced exchange partner for round r by seeding a PRNG with the signature $\langle r, \mathrm{BAL}\rangle_S$. S then deterministically maps the first number generated by the PNRG into the identity of its gossip partner R.

S supplies the seed value, $\langle r, \mathrm{BAL}\rangle_S$, when initiating contact with R. R then verifies that i) the seed is a valid signature, ii) r is the current round, iii) the first number generated by the PRNG when seeded with $\langle r, \mathrm{BAL}\rangle_S$, maps to R, and iv) this is the first time that S has presented this seed value to R. If all four tests pass, then R accepts the gossip request from S.

Note that the selection of R as $S's$ balanced exchange partner in round r is also unpredictable. No other node knows a priori whom S will contact for a balanced exchange in round r because no node can deduce the appropriate digital signature to seed the PRNG.

Using the standard Nash Equilibria analysis, if S assumes all non-Byzantine nodes follow the protocol, then S will clearly not contact a node without the appropriate seed. Likewise, S will not accept an invalid seed from a node Q because Q is by assumption not following the protocol and is therefore Byzantine. S has no incentive to communicate with a node it suspects is Byzantine.

Lemma 1. *In every round, a rational node only communicates with nodes as prescribed by the partner selection algorithm.*

4.1.2 History Exchanges

Problem: *What if a rational node lies about its history?*

After a node S selects a partner R, S and R exchange histories, which consists of three messages. In the first message, S provides a hash of its history and includes the seed value (as discussed earlier) to R; the hash is a verifiable promise to send the corresponding history before learning R's history. After verifying that S should communicate with R by checking the PRNG, R returns its current history. In the final message, S divulges its actual history to R who checks that the previously sent hash is consistent with the divulged history.

Rational nodes might lie about the updates actually in their possession in order to increase their utility. In this paper, we consider rational strategies that seek to increase utility by maximizing the benefit received in each exchange. More sophisticated strategies that maximize benefit across several exchanges—e.g., accepting less benefit in one exchange for increased overall net benefit—are outside the scope of this paper.

Clearly, a rational node has only two options if it chooses to lie about whether it has an update or not: under-reporting or over-reporting. A rational node under-reports an update by disingenuously claiming not to have an update so that it will not have to send it. A rational node will not under-report in a balanced exchange because doing so decreases the expected utility of the exchange in two ways. First, it may limit the exchange to fewer updates than otherwise traded because the partner may not have the particular update. Second, the under-reporting also carries the risk that the partner will send that update that the node already holds but did not report.

A rational node over-reports an update by claiming to possess an update that it does not have. A rational node would do this to gain more utility in an exchange—for example, when its partner has more to offer than would actually be exchanged if the rational node followed the protocol. Note that in order for a rational node to take advantage of its partner, the rational node would have to send a *briefcase* message in which the claimed contents is different from the encrypted contents. We show in Section 4.1.3 that such a briefcase constitutes a proof of misbehavior (POM), which a rational node will never send.

Lemma 2. *A rational node will not over-report its history in either balanced exchanges or optimistic pushes.*

Lemma 3. *A rational node will not under-report its history in balanced exchanges.*

4.1.3 Briefcase Exchange

Problem: *What if a rational node places fake or garbage data in briefcase messages?*

After the history exchange commits S and R to sending k updates that they possess but their partner lacks, each node sends a briefcase message to the other. Each *briefcase* message contains the ids of the two parties, the seed uniquely identifying this exchange, the encrypted updates, and an update list stating what the encrypted contents should be. Furthermore, the sender signs the *briefcase* thereby promising that the encrypted contents are genuine and match the update list.

A rational node might choose to include inappropriate data in briefcase messages to gain benefit without offering anything of value to its partner. We structure *briefcase* messages so that a rational node only sends the expected plaintext fields and encrypted updates. For the plaintext, the receiver knows what to expect in each field and refuses to send the key to its briefcase until it has received a briefcase with a valid label from the partner.

We now explain why encrypted updates are also sent as expected. The key idea is that including updates that do not match the update list in the signed briefcase represent a POM that will lead to a node's eviction. This eviction happens in 3 steps: *i)* if no key is provided or the briefcase contains invalid data, the receiving node will forward the briefcase to the broadcaster, who *ii)* will decrypt the data by generating the appropriate key from its collection of all nodes' private keys and the provided seed, and *iii)* after checking that the briefcase is invalid, issues an eviction notice with the next round of updates.

If a node generates a partner from the PRNG for which it has an eviction notice, the node uses the PRNG to generate the first number that maps to a non-evicted partner. When initiating an exchange, a node provides the eviction notices proving the node's right to avoid its original selected partner.

To provide incentive for reporting invalid *briefcases*, the broadcaster offers small bounties for the first invalid *briefcase* proving a node's guilt. We suggest that the bounty be the inclusion of the reporting node in the set receiving the next update from the broadcaster thereby guaranteeing that the node receives all updates for the next round.

Lemma 4. *Rational nodes never place fake or garbage data in* briefcase *messages.*

Lemma 5. *Rational nodes report malformed* briefcases *to the broadcaster.*

4.1.4 Key Exchange

Problem: *What if a rational node chooses not to send the key or sends an invalid key?*

After a node is satisfied with the plaintext content of the valid signed briefcase it received, the node should send via UDP the key to decrypt the contents of the briefcase it sent.

The problem of exchanging symmetric keys brings to light a hard problem. If we look to the fair exchange impossibility result [39], we see that, in general, there is no deterministic solution to fair exchange without a trusted third party. Surprisingly, we show that in the setting of BAR Gossip, altruistic and rational nodes can exchange keys *fairly enough* without a trusted third party.

Consider the two ways in which a rational node can deviate from the prescribed key exchange phase. First, a rational node may send a key response message containing an invalid key. However, such a signed key response message along with the corresponding briefcase would constitute a POM.

Second, a rational node might ignore its partner's key request messages to save the cost of sending the symmetric key. However, if this rational node believes that its partner will continue resending the key request messages, then the small savings of initially not sending the symmetric key will quickly erode. Therefore, the linchpin in deterring this rational deviation is whether a rational node believes other nodes will resend their key requests.

We use a *credible threat* mechanism to root this belief. A rational node, under the Nash Equilibria assumption, will indeed resend the key request because it believes that its partner is following the protocol, and will thus, faithfully respond.

The key exchange phase therefore provides the following safety and liveness properties.

Lemma 6. *A rational node does not send invalid key response messages.*

Lemma 7. *A rational node eventually responds with a valid key to key request messages.*

4.1.5 Balanced Exchange Discussion

We see that rational nodes faithfully complete each phase of balanced exchanges. Consequenctly, based on Lemmas 1 through 7:

Theorem 1. *Rational nodes participating in a balanced exchange faithfully follow the steps of the Balanced Exchange Protocol.*

4.1.6 Optimistic Push Discussion

The Optimistic Push Protocol follows nearly the same steps as the Balanced Exchange Protocol. As a result, Lemmas 1, 2, 4, 5, 6, and 7 follow directly. However, the extra flexibility to include junk prevents Lemma 3 from applying because a rational node may under-report its young list to minimize the chance of receiving junk.

Additionally, rational nodes may employ a strategy to always send junk instead of useful updates. The intuition behind such a strategy is to maintain the scarcity of updates in that rational node's possession. Although we cannot prove that a rational node would not choose such a strategy, our experiments indicate that the effect of always sending junk actually has a negative impact on a rational node because i) it has no discernible impact on benefit and ii) junk is more expensive to send than legitimate updates.

4.2 Designing for Altruism

We originally intended the Optimistic Push Protocol to provide a way to leverage altruism. In order for a protocol to leverage altruism, a node faithfully executing some part of the protocol should expect a decrease in utility. In the end, we failed to take advantage of altruism and instead designed a better protocol.

At first, we allowed altruistic nodes to offer young updates for free, thus helping the initial dissemination of updates. However, we observed that this was dangerous because rational nodes then would under-report histories in balanced exchanges; rational nodes would gamble on getting an update for free via altruism instead of trading for the same update in a balanced exchange.

To deter this behavior, we introduced the junk updates so that there was no *free ride*. At this point, the optimistic push leveraged altruism because any node executing it received negative utility. By introducing the optimization of replacing the junk updates with something useful—i.e., responding to optimistic pushes with updates from the old list—we changed the expected utility of many optimistic pushes from negative to positive. Indeed, our experiments show that 88% of optimistic pushes never return any junk updates, thus making it desirable for rational nodes to initiate optimistic pushes.

4.3 Designing for Byzantine behavior

Byzantine behavior is an unfortunate reality of distributed systems. In addition to enticing rational nodes to behave correctly, we must also take measures to limit the negative impact that Byzantine participants may have on good users of the system. We limit our attention to attacks within the scope of the protocol; DoS attacks based on network flooding are beyond the scope of this paper.

In traditional Gossip protocols, Byzantine nodes can initiate an arbitrary number of connections and impose arbitrary load on non-Byzantine nodes. Our partner selection criteria limits the number of connections any node can make to a small constant, preventing Byzantine nodes from abusing the system through the creation of arbitrarily many legitimate connections.

Each seed contains only the round and type of exchange. A node can thus generate only two seeds per round, resulting in two communication partners generated from the deterministic PRNG. The inherent limitation on communication prevents a Byzantine node from initiating legimitate connections with more than two nodes in a round. Further, we assume that a node keeps track of the other nodes that have contacted it in the current round to prevent a Byzantine node from legitimately contacting its partners multiple times within a round.

The signatures used during the history, briefcase, and key exchange portions of the protocol prevent rational nodes from sending fake or garbage data. They also provide mechanisms for detecting and verifiably identifying Byzantine nodes that attempt to break the system.

5 Evaluation

In this section, we show that BAR Gossip is a robust p2p streaming protocol capable of providing stable and reliable throughput. We evaluate BAR Gossip through experiments and simulations,[1] and demonstrate that BAR Gossip:

1. Performs better than traditional gossip in the presence of rational nodes (Section 5.3).
2. Prevents unilateral deviation by rational nodes (Section 5.4).
3. Is stable in the presence of significant collusion (Section 5.5).
4. Tolerates up to 20% Byzantine deviation (Section 5.6).

5.1 Methodology

Several parameters characterize the BAR Gossip protocol with regards to the broadcast stream and both Balanced Exchange and Optimistic Push. The broadcaster sends $bcast_{cnt}$ updates per round to $bcast_{hr}$ percent of the audience. The size of an update is, on average, upd_{size} bytes. Members of the audience then exchange the updates until they expire upd_{dl} rounds later. Each round lasts r_{len} seconds. We adopt the Round Retransmission Limit optimization [5], to limit the maximum the number of updates, denoted bal_{size}, that a node is willing to disseminate in a round through Balanced Exchange. The number of updates that a node is willing to disseminate in a given balanced exchange decreases exponentially with the number of balanced exchanges the node has performed so far in that round: the node allocates up to half of bal_{size} updates to its first balanced exchange, half of what's left to the second, and so on. For Optimistic Push, $push_{age}$ denotes the maximum age of updates sent in the young list of the sender, while $push_{size}$ is the maximum length of the receiver's want list. The relative cost of junk, c_{junk}, is the cost of sending a junk update divided by the cost of sending the largest update in the stream. Table 1 provides the values for these parameters for our simulation and prototype experiments. Note that our simulation and prototype experiemnts use a different value of $push_{size}$ in order to maintain the ratio between $bcast_{cnt}$ and $push_{size}$ approximately the same in the two settings.

For our prototype evaluations, we implement the BAR Gossip protocol in Python to stream an MPEG-4 video [40]. We recorded a 220 Kbps UDP video stream at 30 frames per second using Quicktime Broadcaster

[1]Figures denoted with [sim] are derived from simulation data.

Protocol Parameter	Simulation	Prototype
bcast$_{cnt}$(upd.)	10	25–51
bcast$_{hr}$(%)	5	5
upd$_{dl}$(rounds)	10	10
r_{len}(sec.)	1	1
bal$_{size}$(upd.)	*unlimited*	65
push$_{size}$(upd.)	2	8
push$_{age}$(upd.)	3	3
c_{junk}	2	1.27

Experimental Parameter	Simulation	Prototype
# nodes	250	102
upd$_{size}$(bytes)	1024	530–998

Table 1: Protocol parameter settings and experimental settings used in simulations and prototype experiments unless otherwise noted.

with one key frame every 60 seconds. Quicktime Broadcaster generates UDP datagrams for the broadcast with an average size of 764 bytes ($\sigma = 234$), resulting in 25–51 datagrams per second.

Our broadcaster and audience are all 2.5 GHz Pentium 4 machines sharing a 100 Mbit/s Ethernet subnet. The broadcaster reads the recorded video from disk, encapsulates each UDP datagram in an update, and unicasts each update using UDP to a random 5% of nodes. Nodes then exchange updates as in Figure 1. A client delivers an update by extracting the datagram it contains and by sending it to the local Quicktime client that displays the video content. Nodes generate cryptographic hashes using SHA1, and signatures are created using RSA with full domain hashing [4]. The Mersenne Twister algorithm provides a suitable PRNG [30]. During experiments, client machines were underutilized at 30% CPU and 8 MB memory usage. Overall send and receive bandwidth used for exchanges was typically 600 Kbps per client. In the following sections we measure the reliability (expressed as the percentage of updates received by the deadline) jitter (measured as the percentage of rounds in which an update misses its deadline), and bandwidth characteristics of BAR Gossip. Unless otherwise noted, measurements in simulations are averaged over 1000 rounds and using the prototype are averaged over 30 trials. Error bars are small in experimental data and as such are elided from graphs for clarity.

5.2 Traditional Gossip

In the next subsection, we compare BAR Gossip against a traditional push-pull gossip protocol [5], where each node following the protocol selects one partner per round uniformly at random, exchanges histories, and then exchanges missing updates.

In traditional gossip, a rational node will never send an update because there is no benefit gained in the act. Therefore, altruistic nodes do all the work of disseminating updates in the system. Our comparison does not consider rational nodes that initiate more gossip exchanges than prescribed nor does it consider Byzantine behavior because traditional gossip was not designed for such environments.

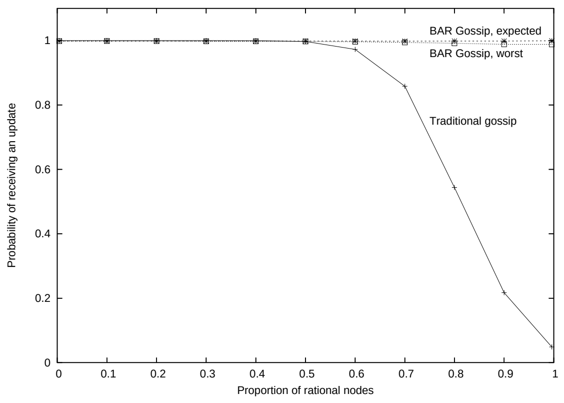

Figure 2: [sim] Reliability experienced by an altruistic node using traditional gossip versus BAR Gossip.

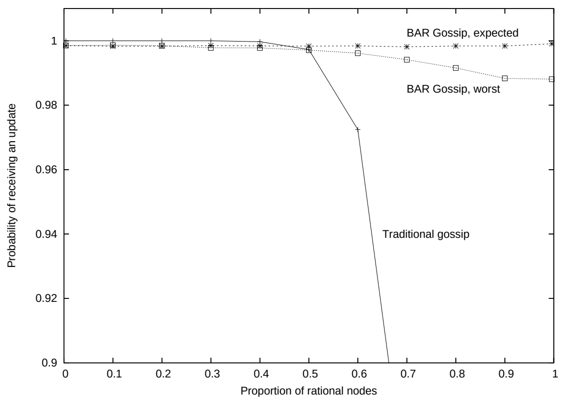

Figure 3: [sim] A closeup of Figure 2 differentiating expected and worst case BAR Gossip.

5.3 BAR Gossip vs. Traditional Gossip

We first show through simulation that while the performance of traditional gossip degrades noticeably as we increase the percentage of rational nodes, the performance of BAR Gossip remains relatively constant. Further, the additional overhead incurred by BAR Gossip is small—a single altruistic node using BAR Gossip among a group of rational nodes incurs only 1% overhead more than an altruistic node using traditional gossip among a group of altruistic nodes with a single rational node.

Figures 2 through 4, plot the reliability and the bandwidth used by an altruistic node as a function of the number of rational nodes.

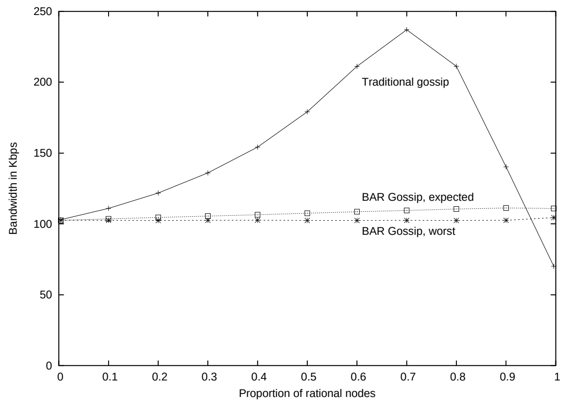

Figure 4: [sim] Send bandwidth used by an altruistic node using traditional gossip versus BAR Gossip.

In traditional gossip, there are no mechanisms to detect which nodes initiated exchanges or how many updates a node (should have) sent. Consequently, rational nodes in a traditional gossip protocol can freeload without directly impacting the quality of service they receive. As Figures 2 through 4 show, however, as more nodes freeload the overall system performance degrades.

The figures contain two lines for BAR Gossip corresponding to the best-case and worst-case strategy that rational nodes can follow from the perspective of an altruistic node. The "BAR Gossip, expected" line is obtained by assuming the best case, in which rational nodes choose to follow the BAR Gossip protocol as specified: they initiate balanced trade and optimistic push exchanges and respond to some other node's optimistic push with useful updates whenever possible. We mark this line "BAR Gossip, expected" because our experimental data, presented in Section 5.4 suggests that the expected behavior of rational nodes in BAR Gossip is indeed to follow the protocol faithfully in its entirety. However, while we have proved in Section 4.1 that rational node will execute the Balanced Exchange protocol, we do not have have a similar proof for Optimistic Push. We thus also show a *worst case* line to bound the range of possible rational behaviors—in the worst case, rational nodes initate no optimistic exchange and respond to optimistic pushes with junk.

Figure 2 demonstrates that an altruistic node in BAR Gossip receives almost all stream updates even when all nodes are rational. In contrast, nodes following the protocol in traditional gossip experience significantly lower reliability once the number of rational nodes exceeds 50%. When all but one node are rational, traditional gossip degenerates to the reliability that can be guaranteed by the broadcaster alone, as rational nodes have no incentive to communicate. Figure 3 highlights an important difference between the expected and worst case behavior for the BAR Gossip protocol. Although the reliability in the worst case is still above 95%, the 2–3% difference between the best and worst case has significant implications for jitter as we will see in the next section.

Figure 4 shows the bandwidth consumed by nodes following the protocol as the fraction of rational nodes varies. The bandwidth performance of the BAR Gossip algorithm is nearly constant, and, in spite of the extra messages required to negotiate the fair and optimistic exchanges, is very close to the bandwidth required by altruistic nodes in the absence of rational node. BAR Gossip succeeds in discouraging free riders, while with traditional gossip altruistic node must bear more of the burden of spreading updates, with bandwidth spiking sharply at 70% of rational nodes, before tumbling down to almost nothing. The dramatic fall off coincides with the sharp decline in reliability in Figure 2. In this area of the graphs, the majority of updates being broadcast are received directly by rational nodes that do not spread them, reducing both reliability and bandwidth devoted to gossipped data.

Note that the rise in bandwidth experienced by altruistic nodes in traditional gossip represents dangerous negative reinforcement: as nodes defect to rational behavior, the remaining altruistic nodes are punished with increased bandwidth load, encouraging them also to defect until reliability collapses. BAR Gossip exhibits robustness to rational behavior with steady reliability and bandwidth measurements.

5.4 Unilateral Rational Deviation

We use now our prototype to examine which strategy an individual rational user would choose if it were to deviate unilaterally from the specified protocol.

In this subsection all nodes but one follow the protocol, while the remaining node explores different strategies. We assume that a rational node will prefer a strategy that improves the quality of the delivered stream by maximizing reliability and minimizing jitter and that minimizing bandwidth is of secondary concern. We make two further observations. First, since we do not consider multi-round strategies, we only examine strategies that provably improve a node's per-round utility. Second, a rational node that is missing one or more updates always has a positive expected benefit from participating in a balanced exchange and, according to Theorem 1, will execute the Balance Exchange protocol faithfully. We then focus our attention on the choices available to a rational node with respct to Optimistic Push. We measure the reliability, bandwidth, and jitter seen by the one adventurous node for each strategy in Table 2.

The *proactive* strategy identifies a rational node that initiates optimistic push exchanges; otherwise, the node is *passive*. The *Data*, *Junk*, and *None* strategies cor-

Strategy	Accepts OP	Initiates OP	Returns
Proactive/Data	Yes	Yes	Data
Proactive/Junk	Yes	Yes	Junk
Proactive/Decline	No	Yes	None
Passive/Data	Yes	No	Data
Passive/Junk	Yes	No	Junk
Passive/Decline	No	No	None

Table 2: The six different strategies that a rational node may follow regarding Optimistic Push Exchanges (OP). The BAR Gossip protocol specifies the proactive/data strategy.

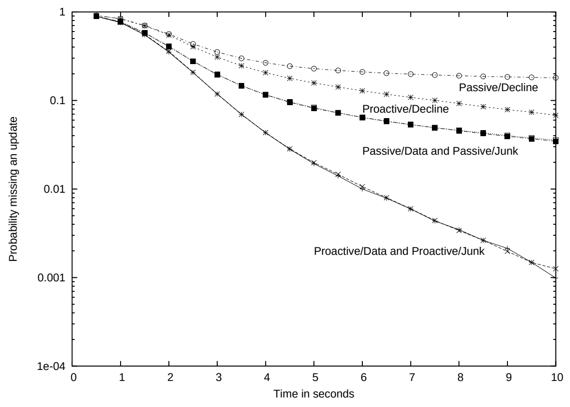

Figure 5: Probability of missing an update experienced by a rational node when all other nodes are following the protocol.

respond to rational nodes responding to an optimistic push exchange with useful updates (if possible), junk, or by declining the exchange, respectively. Note that the proactive/data combination corresponds to a node that follows faithfully the Optimistic Push protocol.

Figure 5 shows the probability that the adventurous node will miss an update by following each of the six strategies, where lower lines correspond to better reliability. Table 3 provides the corresponding average jitter. When taken together, Figure 5 and Table 3 allow us to conclude that rational nodes will prefer either proactive/data or proactive/junk. This is perhaps not surprising, given that proactive strategies perform additional exchanges which are likely to result in more deliverable updates than passive strategies.

Strategy	Avg. Jitter	Std. Deviation
Proactive/Data	0.48%	0.75%
Proactive/Junk	0.52%	1.35%
Proactive/Decline	18.94%	11.50%
Passive/Data	16.61%	6.86%
Passive/Junk	16.39%	4.70%
Passive/Decline	54.55%	6.02%

Table 3: Jitter experienced by a rational node when the remaining nodes are all altruistic.

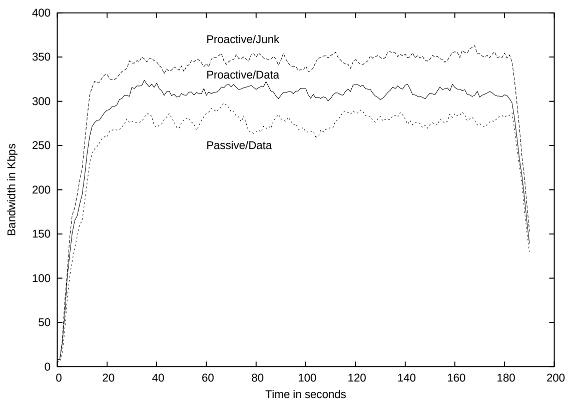

Figure 6: Average send bandwidth used by a rational node when the remaining nodes are all altruistic. Bandwidth is smoothed across the trial using a 4 second sliding average.

The tie breaker between the top two strategies comes from Figure 6, which shows proactive/data to be more bandwith thrifty than proactive/junk. This is not an accident: as Table 1 shows, we have *designed* BAR Gossip with $c_{junk} > 1$ so that rational nodes will prefer filling their briefcases, whenever possible, with valuable updates rather than junk.

The conclusion we draw from this set of experiments is that a rational node, when surrounded by other nodes that follow BAR Gossip, appears to have no obvious incentive to deviate from the protocol—in fact, quite the contrary. While our experiments clearly fall short of proving that BAR Gossip as a whole (Balanced Exchange plus Optimistic Push) constitutes a Nash equilibrium, it does suggest that a Nash equilibrium is likely to be found at or near the strategy that corresponds to BAR Gossip. For instance, while we are unable to prove that there are no beneficial hybrid strategies that, depending on the environment, switch between two or more of the the six strategies we have considered, it appears that the benefit of a proactive strategy derives from consistently participating in more exchanges, making it unlikely that switching occasionally to a passive strategy would provide a net gain. As for switching among proactive strategies, it yields no change in benefit while changing bandwidth costs, also providing little room for improvement.

Overall, we believe that the best (expected) and worst case lines in Figures 2 through 4 provide a reasonable bound on the actual behavior of rational nodes in BAR Gossip.

5.5 Rational Collusion

One of the limitations of Nash equilibria is that they do not model rational behaviors that involve collusion, in which multiple nodes coordinate their actions to maximize their collective utility. We perform a series of simulations to assess the impact that a group of colluding rational nodes may have on the remaining nodes that follow the BAR Gossip protocol.

We simulate a *perfect collusion* scenario in which all colluding nodes broadcast new updates within the group immediately upon receiving an update. Figure 7 shows how the size of a perfect collusion group affects the quality of the stream seen by a node that follows BAR Gossip. In this simulation, the colluding nodes use a passive/decline strategy when gossiping with non-colluding ones.

Note that convergnce is worse than in traditional gossip with rational nodes (see Fig. 2) because updates spread faster among the collusion group than among traditional gossip protocols. Further, while altruistic nodes within traditional gossip perform full exchanges and can thus somehow counterbalance the uncooperative rational nodes, in BAR Gossip the rational nodes outside of the "in-crowd" are limited to perform balanced exchanges.

In small perfect collusion groups, colluding nodes get most of their updates for free from other colluding nodes, reducing their contributions to the rest of the system accordingly. With 1.2% of the nodes in a collusion group (3 of 250), colluding nodes achieve 100% convergence regardless of the strategy used to communicate outside the group. We thus assume that colluding nodes will follow the least-expensive passive/decline strategy. Figure 7 illustrates this strategy's effect on altruistic nodes as the size of the collusion group grows.

We find that when the collusion group size reaches 50% of the participants, altruistic nodes see an average convergence of 93% for an update, resulting in an unusable stream. The perfect collusion group may be feasible for small groups. However, large groups require more bandwidth and latency (to broadcast to all members), eventually degenerating to a broadcast protocol. Ironically, as a colluding group grows in size, it might require BAR Gossip to distribute updates internally as trust begins to break down among members.

Still, collusion amounts toa relatively benign behavior: after all, colluding nodes are not actively disrupting the protocol, but simply are not enthusistically participating in it. It would not be hard to modify BAR Gossip to accommodate groups of nodes that elect to use among themselves their own private dissemination protocol. The broadcaster could provide a skewed PRNG that biases nodes outside the collusion group to gossip with each other and *ii)* skew the initial multicast to reduce redundant messages to the colluding group, increasing the distribution among non-colluding nodes. The colluding group would benefit by receiving fewer unwanted exchange requests and redundant updates and outside nodes would avoid wasting rounds being ignored by colluding members.

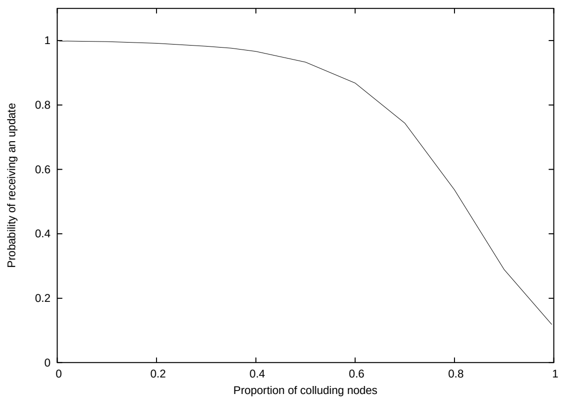

Figure 7: [sim] Effect of collusion on an altruistic node's reliability when colluding nodes are following the passive/decline strategy.

5.6 Byzantine Deviation

Rational players behave according to a well-known utility function and represent most of the nodes in a p2p system across multiple administrative domains. A few Byzantine nodes may possess arbitrary unknown utility functions due to malfunction or maliciousness, possibly affecting rational behavior [1, 33]. Note that Byzantine nodes do not require benefit from their deviations, allowing them to pursue strategies with negative benefit including Denial-of-Service (DoS) attacks.

To assess the robustness of BAR Gossip to Byzantine participation, we assume that the goal of Byzantine nodes is the inverse of any rational player's: to increase the cost and decrease the benefit of all rational nodes. A Byzantine node can, for example, mount a denial of service attack through TCP SYN flooding [23], but such attacks are independent of the details of our specific protocol and are beyond the scope of this paper. We consider Byzantine behavior within the limits of the BAR Gossip protocol. Note that Byzantine nodes cannot compromise safety: all data generated by the broadcaster is signed by the source, and a node is only evicted from the system if there is a signed proof of misbehavior (POM).

We consider worst case Byzantine behavior with respect to a rational node. In the following experiments using our prototype, a Byzantine node always provides a history during a balanced exchange that is the complement of its partner's to induce the other node to exchange the maximum number of updates. During an optimistic exchange, a Byzantine node always announces a complete young list and empty old list if initiating,

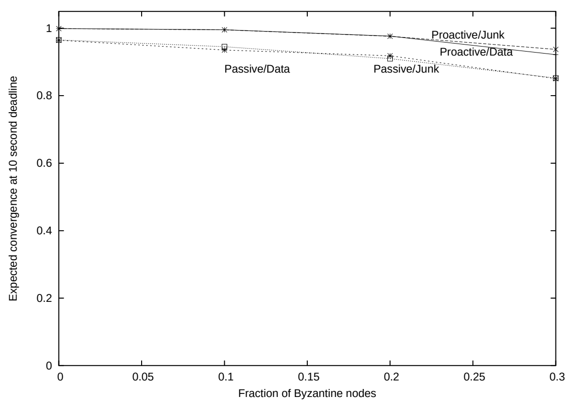

Figure 8: Reliability seen by a rational node employing different strategies while the remaining non-Byzantine nodes follow the BAR Gossip protocol.

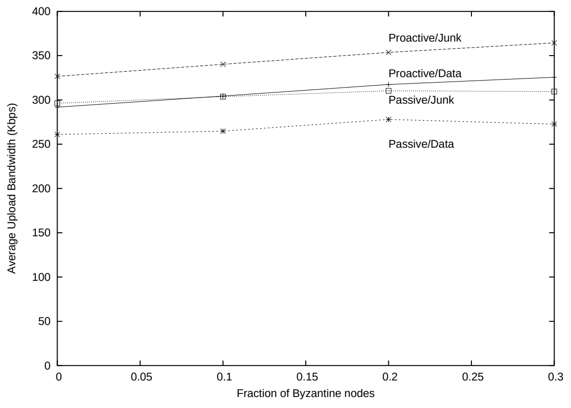

Figure 9: Bandwidth used by a rational node employing different strategies while the remaining non-Byzantine nodes follow the BAR Gossip protocol. Bandwidth is smoothed across the trial using a 4 second sliding average.

and requests the entire young list if receiving. A Byzantine node never sends a briefcase or the corresponding key. In either exchange, the Byzantine behavior does not generate a POM that could evict the node, but the non-Byzantine partner devotes significant bandwidth to the exchange without receiving any benefit. The presence of Byzantine nodes can be viewed as an increase in the overhead associated with the environment as the costs associated with Byzantine nodes depends upon the probability of entering an exchange with a Byzantine node. To mitigate the effects of this Byzantine behavior, rational nodes could skew the likelihood of participating in an exchange with a Byzantine node by maintaining a list of uncooperative nodes, refusing some proportion of connections to such nodes. To show worst case behavior, non-Byzantine nodes in our experiments ignore previous unproductive exchanges. We ignore proactive/decline and passive/decline strategies in which rational nodes decline to participate in optimistic push exchanges.

Figures 8 and 9 show the reliability seen and bandwidth used, respectively, by a rational node pursuing each strategy in the presence of different proportions of Byzantine nodes. The remaining non-Byzantine nodes are altruistic. The choice of strategies remain similar to Section 5.4 where we considered unilateral deviation with no Byzantine nodes. When the proportion of Byzantine nodes grows to 10% passive strategies do not generate a viewable video stream while proactive strategies fall apart at 30% Byzantine nodes. In either strategy, sending valid updates is cheaper than sending junk in optimistic push exchanges.

We conclude that among the strategies available, a rational node should follow the protocol (proactive/data) regardless of the presence of Byzantine nodes. If all non-Byzantine nodes are altruistic, with a system comprised of 20% Byzantine nodes, the bandwidth costs increase by less than 10% while the convergence suffers by only 3%. However, when the proportion of Byzantine nodes grows to 30%, the video stream becomes unusable at 92% convergence.

5.7 Topology Awareness

One obvious way in which our prototype can be improved is by considering network proximity in partner selections. Several studies have demostrated the advantages of proximity-aware gossip protocols [12, 19, 25].

The challenge in applying these techniques to BAR Gossip is in limiting the non-determinism. BAR Gossip's partner selection algorithm uniformly maps pseudo-random numbers to partners. To consider network proximity, we need to bias this mapping, yet retain the verifiable and unpredictable qualities that BAR Gossip's partner selection algorithm currently guarantees. One possible approach would be to globally assign Vivaldi [11] coordinates to every node and use the Vivaldi distance between the nodes as an estimate of their actual distance. Each node can then cluster its neighbors into groups, as in [19] and use the PRNG to bias peer selection towards nodes in closeby groups.

6 Conclusion

We present the first peer-to-peer data streaming application with predictable throughput and low latency in the presence of correct, selfish, and malicious nodes. At the core of our application is BAR Gossip, the first gossip protocol defined under the BAR model. We leverage a unique signature scheme to generate verifiable pseudo-random numbers, allowing us to eliminate the opportunity for rational nodes to hide behind nondeterminism without sacrificing the benefits of the random communication pattern of gossip. Our experiments and simulations show that our protocol provides good convergence properties as long as no more than 20% of the nodes are Byzantine or no more than 40% of the nodes collude. In both cases, nodes following the protocol receive more than 95% of the relevant updates in less than 10 seconds.

References

[1] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. Bar fault tolerance for cooperative services. In *Proc. 20th SOSP*, Oct. 2005.

[2] G. Badishi, I. Keidar, and A. Sasson. Exposing and eliminating vulnerabilities to denial of service attacks in secure gossip-based multicast. In *DSN '04*, page 223, Washington, DC, USA, 2004. IEEE Computer Society.

[3] M. Balakrishnan, S. Pleisch, and K. Birman. Slingshot: Time-critical multicast for clustered applications. In *IEEE Network Computing and Applications*, 2005.

[4] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *CCS '93*, pages 62–73, New York, NY, USA, 1993. ACM Press.

[5] K. P. Birman, M. Hayden, O. Oskasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM TOCS*, 17(2):41–88, May 1999.

[6] K. P. Birman, R. van Renesse, and W. Vogels. Spinglass: Secure and scalable communications tools for mission-critical computing. In *DARPA DISCEX-2001*, 2001.

[7] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: high-bandwidth multicast in cooperative environments. In *Proc. 19th SOSP*, pages 298–313. ACM Press, 2003.

[8] R. Chandra, V. Ramasubramanian, and K. Birman. Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks. Technical report, Ithaca, NY, USA, 2001.

[9] B. Cohen. The bittorrent home page. http://bittorrent.com.

[10] B. Cohen. Incentives build robustness in bittorrent. In *Proc. 2nd IPTPS*, 2003.

[11] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM '04*, pages 15–26, New York, NY, USA, 2004. ACM Press.

[12] A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proc. of PODC '87*, Aug. 1987.

[13] A. K. Dixit and S. Skeath. *Games of Strategy*. W. W. Norton & Company, 1999.

[14] J. R. Douceur. The Sybil attack. In *Proc. 1st IPTPS*, pages 251–260. Springer-Verlag, 2002.

[15] P. Eugster, S. Handurukande, R. Guerraoui, A. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. In *Proc. of DSN '01*, July 2001.

[16] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and whitewashing in peer-to-peer systems. In *Proc. PINS*, pages 228–236. ACM Press, 2004.

[17] A. J. Ganesh, A.-M. Kermarrec, and L. Massouli. Peer-to-peer membership management for gossip-based protocols. *IEEE Trans. Comput.*, 52(2):139–149, 2003.

[18] I. Gupta, K. Birman, and R. van Renesse. Fighting fire with fire: using randomized gossip to combat stochastic scalability limits, 2002.

[19] I. Gupta, A. Kermarrec, and A. Ganesh. Efficient epidemic-style protocols for reliable and scalable multicast. In *Proc. of SRDS '02*, 2002.

[20] A. Habib and J. Chuang. Incentive mechanism for peer-to-peer media streaming. In *12th IEEE International Workshop on Quality of Service.*, 2004.

[21] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03*, 2003.

[22] S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of non-repudiation protocols. *Computer Communications*, 25(17):1606–1621, Nov. 2002.

[23] J. Lemon. Resisting syn flood dos attacks with a syn cache. In *Proceedings of the BSDCon 2002*, pages 89–97, Berkeley, CA, USA, 2002. USENIX Association.

[24] D. Levin, R. Sherwood, and B. Bhattacharjee. Fair file swarming with fox. In *IPTPS*, Feb 2006.

[25] M.-J. Lin and K. Marzullo. Directional gossip: Gossip in a wide area network. In *European Dependable Computing Conference*, pages 364–379, 1999.

[26] D. Malkhi, Y. Mansour, and M. K. Reiter. Diffusion without false rumors: on propagating updates in a byzantine environment. *Theor. Comput. Sci.*, 299(1-3):289–306, 2003.

[27] D. Malkhi, E. Pavlov, and Y. Sella. Optimal unconditional information diffusion. In *Proc. of DISC '01*, pages 63–77, 2001.

[28] D. Malkhi, M. Reiter, O. Rodeh, and Y. Sella. Efficient update diffusion in byzantine environments. In *Proc. 20th SRDS*, 2001.

[29] S. Marti and H. Garcia-Molina. Identity crisis: Anonymity vs. reputation in p2p systems. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, 2003.

[30] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998.

[31] R. Melamed and I. Keidar. Araneola: A scalable reliable multicast system for dynamic environments. In *Proc. of NCA '04*, pages 5–14, Washington, DC, USA, 2004.

[32] Y. Minsky and F. Schneider. Private communication.

[33] T. Moscibroda, S. Schmid, and R. Wattenhofer. When selfish meets evil: Byzantine players in a virus inoculation game. In *Proc. PODC '06*, 2006.

[34] A. Nandi, T.-W. J. Ngan, A. Singh, P. Druschel, and D. S. Wallach. Scrivener: Providing incentives in cooperative content distribution systems. In *Proceedings of the ACM/IFIP/USENIX 6th International Middleware Conference (Middleware 2005)*, Grenoble, France, Nov. 2005.

[35] J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54:286–295, Sept 1951.

[36] National Institute of Standards and Technology. *FIPS PUB 186-2: Digital Signature Standard (DSS)*. Jan. 2000.

[37] T.-W. Ngan, D. S. Wallach, and P. Druschel. Incentives-compatible peer-to-peer multicast. In *2nd Workshop on Economics of Peer-to-Peer Systems*, 2004.

[38] D. C. Oppen and Y. K. Dalal. The clearinghouse: a decentralized agent for locating named objects in a distributed environment. *ACM Trans. Inf. Syst.*, 1(3):230–253, 1983.

[39] H. Pagnia and F. C. Gärtner. On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Darmstadt University of Technology, Department of Computer Science, Darmstadt, Germany, Mar. 1999.

[40] A. Puri and A. Eleftheriadis. Mpeg-4: an object-based multimedia coding standard supporting mobile applications. *Mob. Netw. Appl.*, 3(1):5–32, 1998.

[41] R. V. Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. Technical report, Ithaca, NY, USA, 1998.

[42] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Commun. ACM*, 43(12):45–48, 2000.

[43] L. Rodrigues, S. Handurukande, J. Pereira, R. Guerraoui, and A. Kermarrec. Adaptive gossip-based broadcast, 2003.

[44] R. van Renesse, K. P. Birman, D. Dumitriu, and W. Vogels. Scalable management and data mining using astrolabe. In *Proc. of IPTPS '01*, pages 280–294, London, UK, 2002. Springer-Verlag.

[45] R. van Renesse, H. Johansen, and A. Allavena. Fireflies: Scalable

support for intrusion-tolerant overlay networks. In *EuroSys '06*, 2006.

[46] W. Vogels, R. van Renesse, and K. Birman. The power of epidemics: robust communication for large-scale distributed systems. *SIGCOMM Comput. Commun. Rev.*, 33(1):131–135, 2003.

[47] X. Zhang, J. Liu, B. Li, and T. P. Yum. Donet/coolstreaming: A data-driven overlay network for live media streaming. In *IEEE INFOCOM*, Mar. 2005.