Harry C. Li, Allen Clement, Edmund L. Wong,
Jeff Napper, Indrajit Roy,
Lorenzo Alvisi, Michael Dahlin

BAR Gossip

Laboratory for Advanced Systems Research (LASR),
Dept. of Computer Sciences,
The University of Texas at Austin

Introduction

Problem

- Increasing popularity of streaming live events
- Hundreds of thousands of viewers all over the world
- Content delivery in a timely manner to provide satisfying user experience

Possible solution

- Peer-to-peer (p2p) streaming method
- Potential to be highly robust, scalable and adaptive
- Large-scale content providers may adopt p2p-based solutions to shift costs (like bandwidth) to clients
- Small-scale providers might find it simpler to use a self-organizing p2p network instead of provisioning and maintaining a large dedicated server

Prerequisites

- Guarantee of highly reliable, stable, and timely throughput of messages despite the presence of faulty, misconfigured, or even malicious peers
- Robust against selfish users, who try to catch a *free ride* by receiving streams without contributing their fair share to other users

Intriguing implementation

- **BAR Gossip**: p2p data streaming application
- Robust, scalable and reliable data dissemination
- *Verifiable pseudo-random partner selection*
- *Fair enough exchange*
- Predictable throughput and low latency in the **BAR model**

Model

BAR model

- **B**yzantine - behave arbitrarily or according to some unspecified utility function
- **A**ltruistic - follow the protocol as given regardless of costs
- **R**ational - follow a strategy that maximizes its utility

Assumptions

- Clients subscribe to the live broadcast prior to its start and remain in the system for the duration of the broadcast
- Each participant (IP address) is limited to one identity
- Each private key generates unique signatures:
  for a given m, there must exist exactly one valid signature of m by i

POM

- Proof of misbehavior (POM) - sequence of signed messages that proves a client sent a message inconsistent with the protocol specification
- A POM against a client is sufficient evidence to evict that client from the population

BAR Gossip Design

Required properties

- Non-Byzantine clients do not deliver unauthentic stream packets
- Every altruistic client receives a large fraction of all stream packets in a timely manner

Arrangement

- Each client generates a session key pair consisting of a public and private key and signs up for the event by divulging **both** keys to the broadcaster
- The broadcaster then verifies the keys, closes the sign up service and posts a list that contains each client’s identity, address and public key

Broadcasting process

- The broadcaster divides the stream into discrete fixed-size chunks that we call *updates*
- BAR Gossip is a sequence of rounds of duration $T + \delta$ where updates are sent by the broadcaster and exchanged among clients
- Each update expires *deadline* rounds after it was sent by the broadcaster

Broadcasting process

- In each round, the broadcaster multicasts *ups_per_round* updates to subsets of clients
- For each update the broadcaster selects *nSeeds* random clients to receive the update, signs the update and multicasts it to the selected clients
- A client is unlikely to receive all updates directly from the broadcaster and relies on two protocols to garner the remaining: **Balanced Exchange** and **Optimistic Push**

Balanced Exchange

- In balanced exchanges, each party: Sender (S) and Receiver (R) determines the largest number of new updates it can exchange while keeping the trade equal
- A client concurrently initiates an exchange with another client and responds to Balanced Exchange requests from other clients

Balanced Exchange: Phases

- Partner selection
- History exchange
- Update exchange
- Key exchange

Balanced Exchange: Properties

- **Theorem:** If two rational clients participating in a balanced exchange with each other seek to maximize the utility of that exchange independent of concurrent or future exchanges, then following the Balanced Exchange Protocol is a Nash equilibrium
- **Lemma:** A rational client S never issues a POM against itself and expects no benefit from communicating with evicted clients

Balanced Exchange: Partner Selection

- A client S generates an unpredictable, deterministic seed for the pseudo-random number generator (PRNG) to select a random partner, who can then verify and accept the selection using the same PRNG or reject it otherwise
- S then deterministically maps numbers generated by the PRNG to client ids until it finds the first partner R for which it does not have an eviction notice
- R then determines whether the seed is valid

Balanced Exchange: History exchange

- S provides in the first message a hash of its history H_S and the PRNG seed value to R
- After verifying that S is entitled to communicate with R, R returns its current history H_R
- In the final message, S divulges its actual history, H_S, to R who checks that the previously sent hash is consistent with the divulged history

Balanced Exchange: Update exchange

- S and R send the corresponding updates contained in signed briefcases
- Each sender signs the briefcase, promising that the encrypted contents match the description

Balanced Exchange: Key exchange

- The client sends via UDP a key request containing the exchange seed and responds to key requests (also via UDP) with a signed response that contains the seed value and the decryption key corresponding to the briefcase sent in the previous phase
- A client repeatedly sends key requests, up to some constant number of times, until it obtains a key response from its partner

Optimistic Push

- Provides a safety net for clients who have fallen behind by allowing clients to obtain missing updates without giving back a set of updates of equivalent value
- Like the Balanced Exchange except the history and update exchanges

Optimistic Push: 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 c updates from the young list that R is actually missing

Optimistic Push: Update exchange

- S's briefcase contains the c updates from the want list with an appropriate plaintext description of the update ids
- R's briefcase also contains c updates, but the plaintext description does not identify the particular updates, only that c items are inside each of which can be either from the old list or *junk* (crafted to be **larger** than real update) and at least one of those updates is from the old list

Balanced Exchange and Optimistic Push Protocols

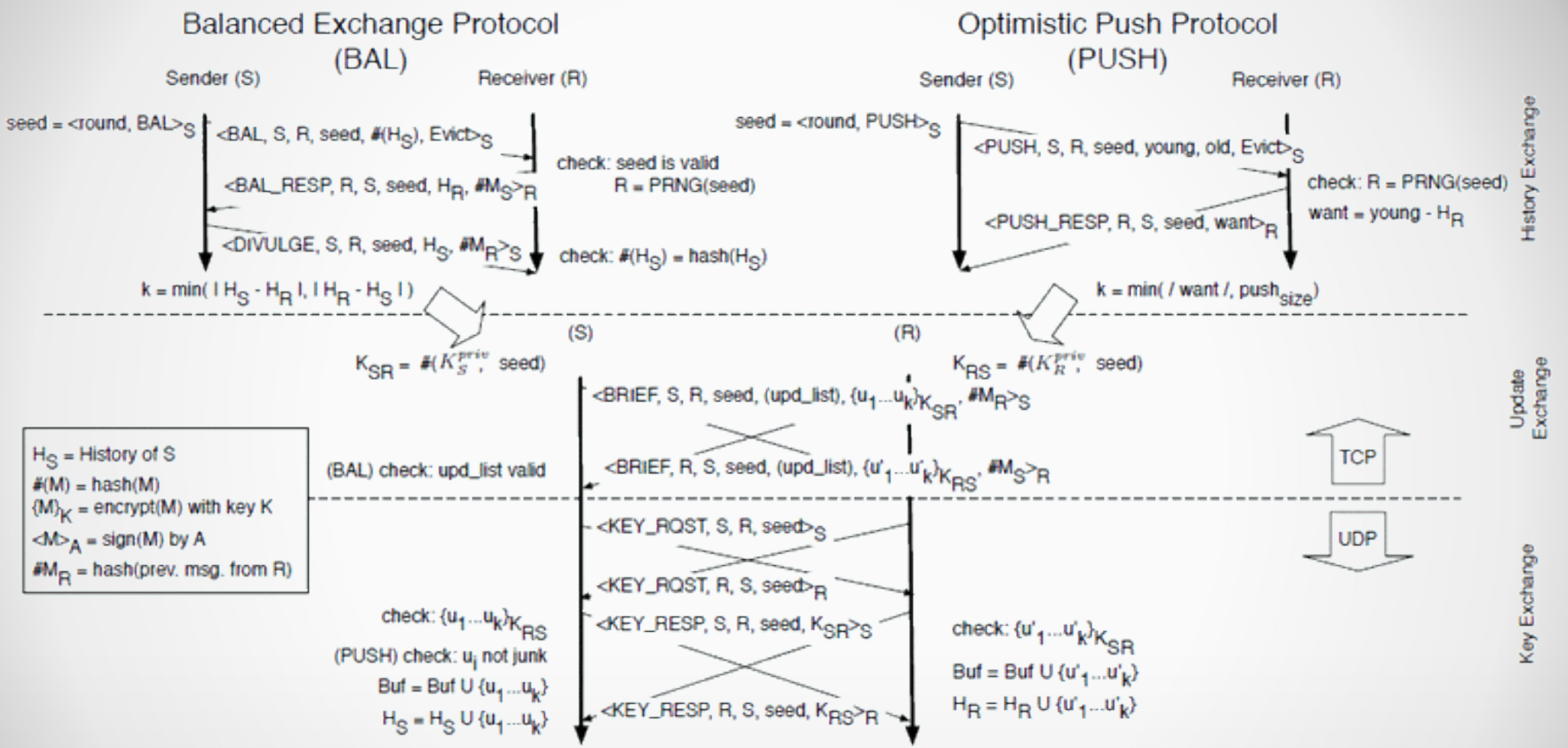

Figure 1: Balanced Exchange and Optimistic Push Protocols for node A contacting node B. During the update exchange, the upd_list is sent in the Balanced Exchange protocol, but omitted for Optimistic Push.

System's auditor

- A trusted agent of the broadcaster to audit possible POMs
- If a queried client does not have a POM against a peer, then the client must reply with a dummy message
- The auditor treats clients that ignore audit requests as it would clients that have provably misbehaved
- The auditor evicts a misbehaving client by sending a signed eviction notice to the broadcaster

Designing for Byzantine behavior

- In BAR Gossip, Byzantine nodes cannot subvert the system’s safety properties because the broadcaster signs each update
- Peer selection protocol limits the number of nodes that one can contact in a round
- A Byzantine client can remain silent during an exchange to slow the spread of updates, but fortunately, gossip protocols are naturally resilient to crash failures

Evaluation

BAR Gossip capabilities

- Outperforms traditional gossip in the presence of rational clients
- Prevents unilateral rational deviation
- Is stable in the presence of significant collusion
- Tolerates up to 20% of the clients being Byzantine

Parameter settings

Protocol Parameter	Simulation	Prototype
ups_per_round(updates)	10	98-101
nSeeds(clients)	25	3
deadline(rounds)	10	10
push_size(updates)	2	20
push_age(updates)	3	3
junk_cost	2	1.39
# clients	250	45

Table 1: Parameter settings used in simulations and prototype experiments.

Gossip algorithms comparison

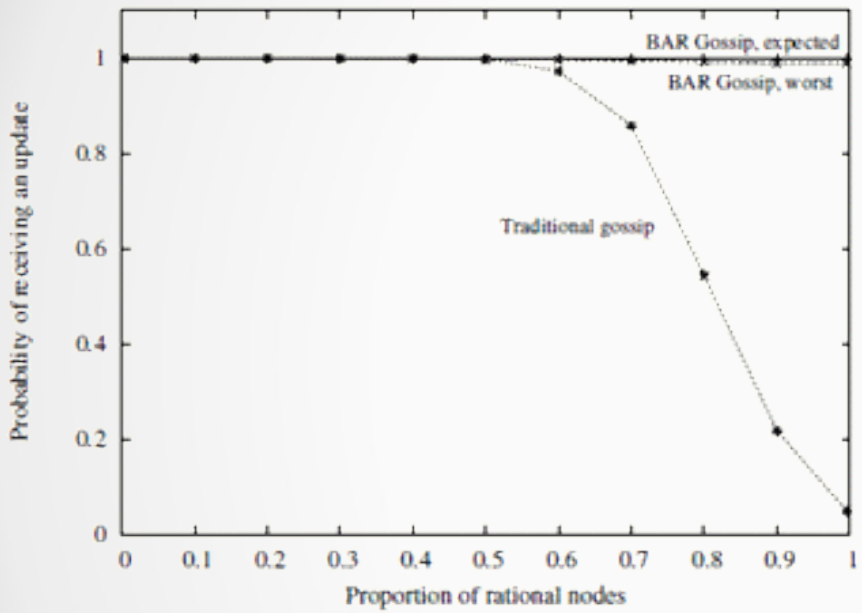

Figure 2: [sim] Reliability experienced by an altruistic client using traditional gossip versus BAR Gossip.

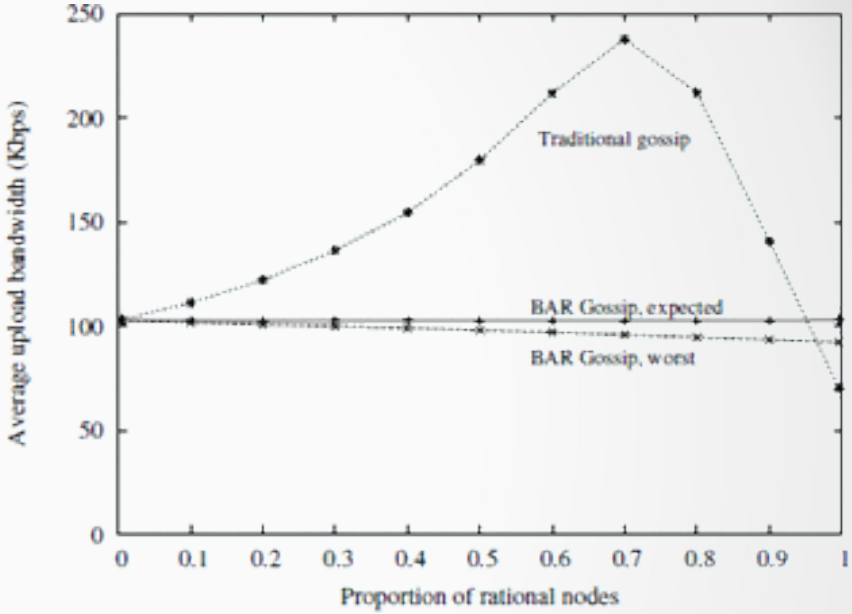

Figure 3: [sim] Send bandwidth used by an altruistic client using traditional gossip versus BAR Gossip.

Rational strategies: Characteristics

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: Six strategies a rational client may follow with regards to the Optimistic Push Protocol.

Strategy	Avg. Jitter	Std. Deviation
Proactive/Data	0.48%	1.16%
Proactive/Junk	0.32%	0.78%
Proactive/Decline	11.59%	6.22%
Passive/Data	18.10%	6.08%
Passive/Junk	14.76%	9.44%
Passive/Decline	47.94%	7.52%

Table 3: Rational client's experienced jitter pursuing different strategies.

Rational strategies: Comparison

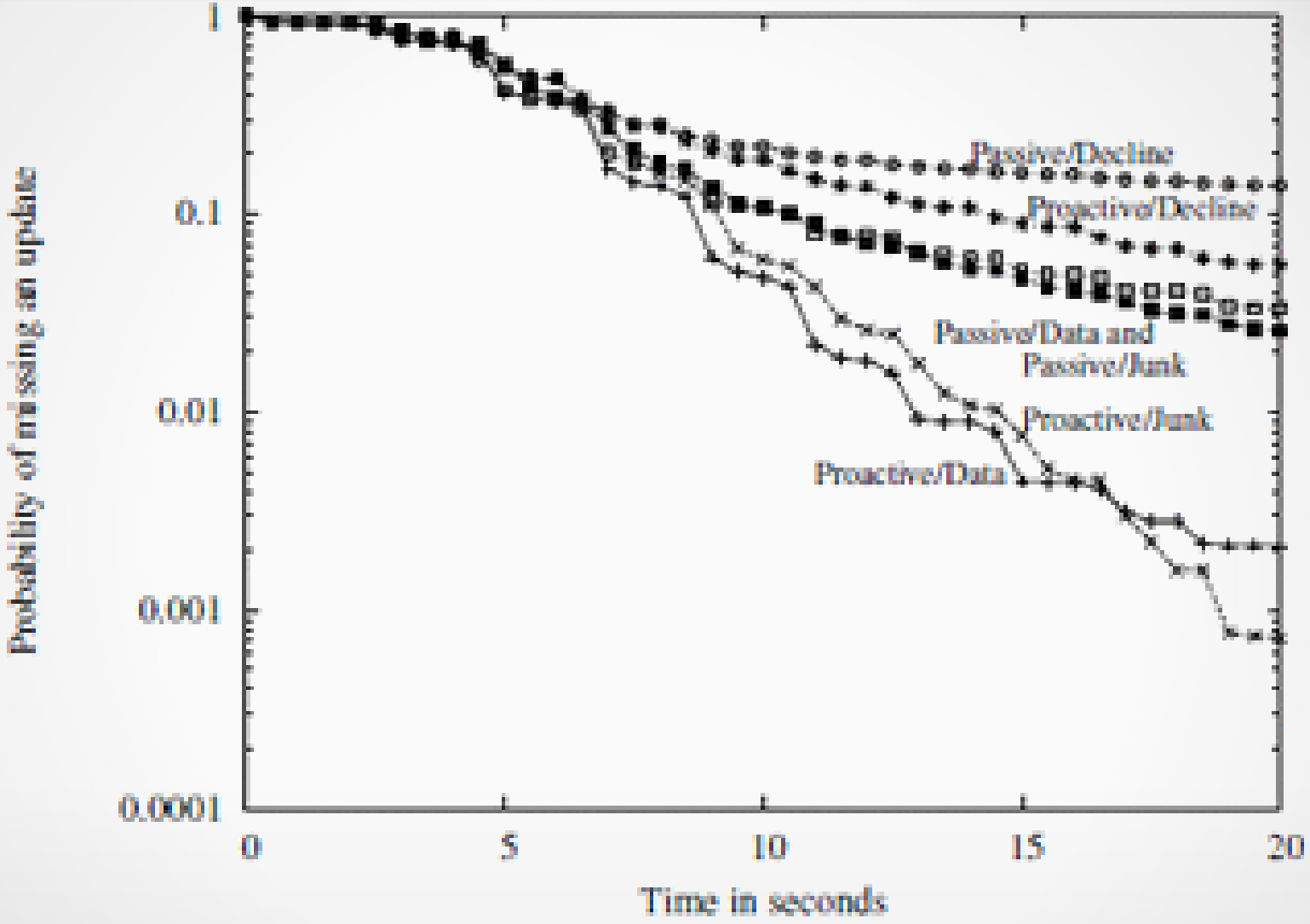

Figure 4: Probability of a rational client missing an update for different strategies.

Rational strategies: Tie breaker

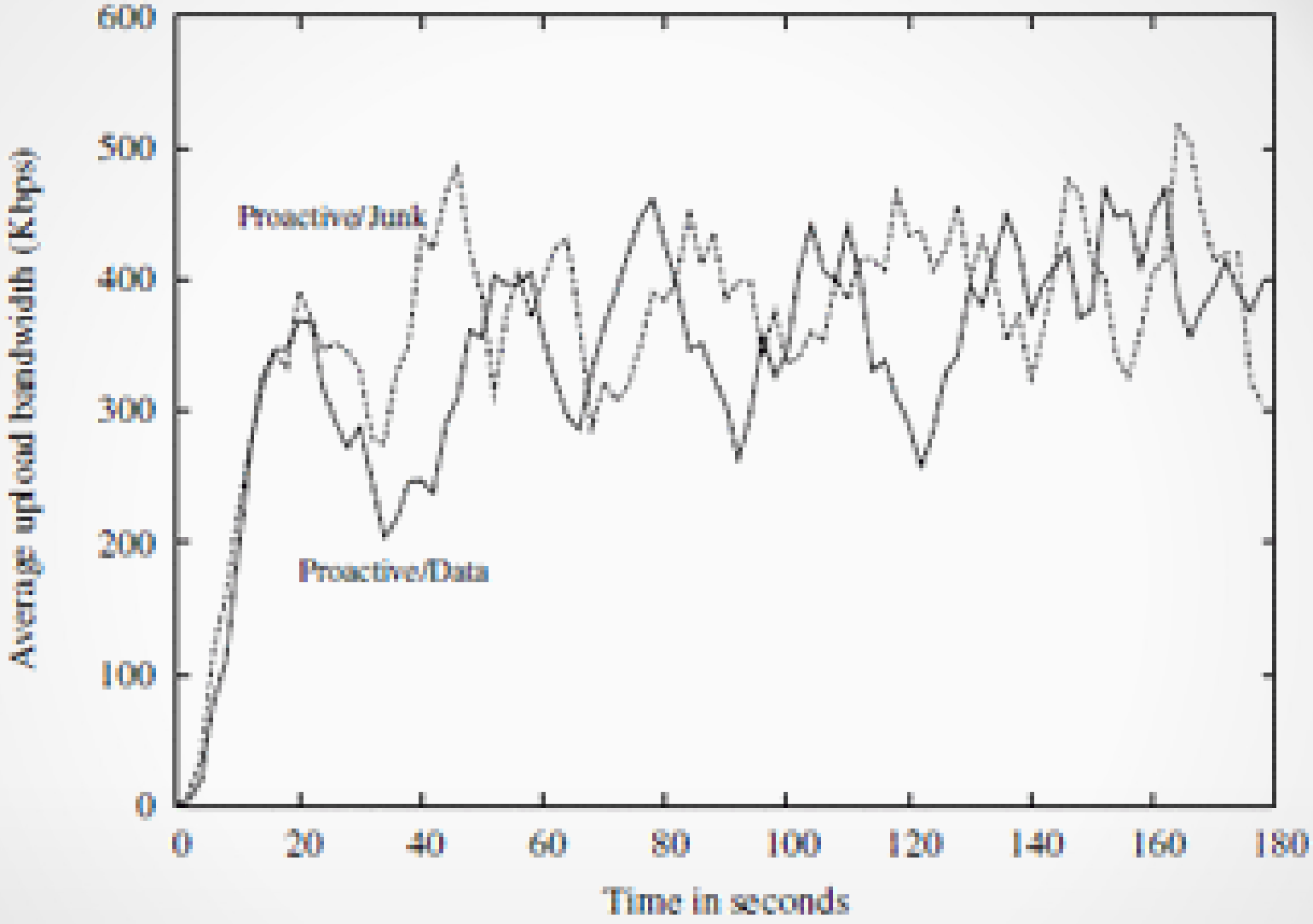

Figure 5: Rational client's consumed bandwidth for different strategies.

Related Work

BitTorrent

- File-sharing p2p application
- Has recognized the issue of a free ride and introduced a set of heuristics to incentivize faithful participation
- Leverages a local reputation scheme for file-sharing in which nodes give preference to those peers who have reliably reciprocated in the past

Scrivener

- 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 unpopular

FOX

- Guarantees optimal download time to all the nodes interested in acquiring the same file under the assumption that all nodes are selfish
- This guarantee, however, comes at the expense of robustness
- The system's incentive structure depends on the fear of mutual assured destruction and a single Byzantine node can cause the entire system to collapse

Splitstream

- A tree-based multicast protocol that achieves load balancing by dividing content into multiple stripes, each of which is multicast using a separate tree
- If Splitstream's multicast trees are periodically rebuilt and nodes maintain local reputations regarding nodes that have misbehaved, then upstream nodes in a given tree have an incentive to provide good service to downstream nodes to avoid future retaliation