Discouraging Selfishness in Lossy Peer-to-Peer Networks

ALEXANDER FRIEDMAN

Discouraging Selfishness in Lossy Peer-to-Peer Networks

Final Paper

In Partial Fulfillment of the Requirements
for the Degree of Master of Science in Electrical Engineering

ALEXANDER FRIEDMAN

Submitted to the Senate of the Technion – Israel Institute of Technology
Shvat 5769 HAIFA February 2009

The Final Paper Was Done Under the Supervision of Associate Professor Idit Keidar in the Department of Electrical Engineering, Technion

THE GENEROUS FINANCIAL HELP OF THE TECHNION IS GRATEFULLY ACKNOWLEDGED

Acknowledgments

I would like to thank my advisor, Associate Professor Idit Keidar, for enriching my knowledge, reading endless drafts, and being always open for questions. Her guidance, patience, invaluable assistance, and generous offers of support and expertise helped me to make this research thorough.

I would like to thank my girlfriend, Rinat, for her support and encouragement during the years.

And last, but not least, I would like to thank my family and friends, who were supportive and understanding at all times.

Contents

List of Figures

List of Algorithms

Abstract

We present *Loss-Tolerant Selfishness Monitor (LTSM)*, a generic service for detecting selfish behavior in various Peer-to-Peer (P2P) applications, such as Mobile Ad-hoc Network (MANET) routing and data streaming (multicast). Resources in P2P systems are provided by the participating peer nodes themselves; each node has to contribute memory, CPU power, bandwidth, and energy. Since most nodes in a MANET are battery-powered, energy is a scarce resource in such an environment. In commercial P2P applications, nodes may exhibit selfish behavior by tampering with the P2P protocol in order to lower their cost. Consequently, it is important for such protocols to work well even when users are equipped with a selfish version of the protocol.

Unlike most previous selfishness-resistant protocols, LTSM can be used in networks subject to message loss, where selfish behavior detection is particularly challenging. For example, wireless networks, such as MANETs, inherently suffer from high packet loss rates. Furthermore, multicast systems for streaming video or audio typically use unreliable transport like UDP, since it is acceptable for some of the data to be lost.

One of our main contributions is mathematically analyzing the impact of various system parameters on the incentives for cooperation, and showing how to choose these parameters so as to ensure that full cooperation is a Nash Equilibrium, at a minimal cost. We illustrate the applicability of LTSM in two exemplar contexts: multicast and MANET routing.

List of Symbols

UDP	User Datagram Protocol
LTSM	Loss-Tolerant Selfishness Monitor
MANET	Mobile Ad-hoc Network
P2P	Peer-to-Peer
DSR	Dynamic Source Routing
CBR	Constant Bit Rate
TFT	Tit For Tat
W	Window Size
τ	Detection Threshold
X	Number of packets sent in a window
$D(X, \tau)$	Detection probability function
$\epsilon(\tau)$	False positive probability function
$Ecost(X, \tau)$	Expected cost
$f(\tau, W, n)$	Lower bound on fine
Γ	Gamma function
$I_z(a, b)$	Regularized incomplete beta function
δ	False positive probability
$f_{max}(\delta, W)$	Lower bound on fine

Chapter 1

Introduction

Peer-to-Peer (P2P) protocols are used in numerous different settings, e.g., P2P multicast systems, file sharing networks, and mobile-ad-hoc networks (MANETs). The underlying networks used by many such P2P systems are lossy. For example, wireless networks, such as MANETs, inherently suffer from high packet loss rates. Furthermore, multicast systems for streaming video or audio typically use unreliable transport like UDP, since it is acceptable for some of the data to be lost.

Resources in P2P systems are provided by the participating peer nodes themselves; each node has to contribute memory, CPU power, bandwidth, and energy. For example, the high bandwidth demand by P2P nodes is driving Internet Service Providers (ISPs) to implement a tiered pricing scheme, where high tier pricing schemes allow unlimited transfers, and lower-tier pricing schemes charge for excess usage [35]. Energy is a scarce resource for battery-powered nodes, such as laptops and PDAs. In commercial P2P applications, nodes may exhibit selfish behavior by tampering with the P2P protocol in order to lower their cost [4, 11, 26]. Consequently, it is important for such protocols to work well even when users are equipped with a selfish version of the protocol.

In recent years, much research has been dedicated to tackling selfish behavior in various P2P applications (e.g., multicast, file sharing, and MANET routing, – see Chapter 2). Many challenging issues, however, remain open. Previous work, for instance, has not exposed and leveraged the similarity among different P2P protocols. Rather, each previous work has focused on one specific protocol, in one specific setting. Another challenge

largely overlooked in previous work is lossy networks (with the exception of [40, 41] – see Chapter 2). Selfish behavior detection becomes much more challenging when one has to cope with unpredictable packet loss. Conventional detectors, such as those used in [3, 5, 26, 28], would wrongfully accuse cooperating nodes for not sending lost packets. Finally, previous work has not mathematically quantified the relationship that needs to hold among system parameters such as a cooperating node's cost, the penalty for lack of cooperation, and the decision when to punish a node, in order to make full cooperation a Nash Equilibrium at a minimal cost.

In this thesis, we address the three open issues above. We leverage the similarity among many different P2P protocols in order to define a common monitoring abstraction suitable for detecting selfish behavior in various such protocols (Chapter 3). Our abstraction's interface enables each peer node to monitor other nodes, and to determine when to punish a node for alleged misbehavior. We then present a loss-tolerant selfishness monitor (LTSM) that implements this interface (Chapter 4). When using LTSM, a node blamed for lack of cooperation must pay a fine to continue participating in the protocol.

One of the main contributions of this thesis is mathematically quantifying the relations between the above fine, the cost of performing a basic operation (such as sending a message), the packet loss rate, the decision as to when to punish a node, and the costs incurred by cooperative and non-cooperative nodes (Chapter 4.1). We show how to tune LTSM's parameters so as to make full cooperation and fully following the protocol a Nash Equilibrium, while minimizing the cost for cooperating nodes.

To illustrate the applicability of our abstraction, we show (Chapter 5) how it can be seamlessly employed in existing multicast schemes [9, 21, 30], and in existing schemes for MANET routing [3, 5, 18, 26, 28]. By using LTSM with the appropriate parameter settings in these applications, one can automatically make them robust to packet loss.

Chapter 2

Related Work

Selfish behavior has been widely studied in various P2P systems, e.g., content distribution protocols [11, 16], tree-based multicast [30], gossip-based multicast [21, 24], and MANET routing [1, 2, 3, 5, 6, 7, 8, 12, 18, 26, 28, 29, 34, 37, 39, 40, 41, 42].

2.1 Content Distribution and Multicast

The BitTorrent [11] and Avalanche [16] P2P content distribution protocols rely on a Tit-For-Tat (TFT) strategy to encourage participation and discourage selfish behavior. In this strategy, a user prefers to upload data blocks to users from which it currently downloads some other blocks.

Ngan et al. [30] propose a tree-based multicast protocol, based on SplitStream [9], which detects selfish nodes, and periodically reconstructs trees to prune misbehaving nodes. In their scheme, each node maintains a debt counter for every other peer it encounters. This counter is equal to the number of packets forwarded to the peer minus the number of packets received from the peer. A rational node does not allow this debt to grow beyond a predefined threshold. The debt counters are expected to remain balanced, on average, by periodically reconstructing the multicast tree, such that each new tree is sufficiently different from the previous tree. A trade-off exists between the overhead of tree reconstruction, and the time it takes to detect a non-cooperative node. The authors also revisit SplitStream's tree construction phase, in which a node may selfishly refuse

to accept a child in the pretense of serving other children. They suggest that a node that frequently refuses to accept a child is likely to be non-cooperative.

Habib and Chuang [17] propose an incentive-based media streaming protocol. In their protocol, the quality of service each node enjoys depends on its level of cooperation. Namely, peers are ranked according to their cooperation. Each peer may receive service only from nodes whose ranks are lower than its own. Thus, the peer selection process takes relative contributions into account.

The authors of the EquiCast [21] gossip-based multicast protocol employ game theory to formally prove cooperation when all participating nodes are selfish and rational. In gossip protocols, nodes exchange data with randomly selected peers. Similarly to [30], each node maintains a balance counter for each of its neighbors in the mesh overlay. Unlike in [30], however, the counter inherently remains balanced, in average, by using gossip over a mesh-based overlay, instead of a multicast tree. A rational node disconnects its link with a node if that node's balance is lower than a predefined negative threshold. Special fine packets, which do not affect the balance, are used to punish nodes with negative balances which are higher than the above threshold. Failing to send such fine packets constitutes eviction. The multicast source is used to help in the recovery of cooperative nodes, whose balance becomes negative due to some unfortunate circumstances, by sending data packets to the negatively balanced peer in return for sending the same number of fine packets. The balance is also bound by a threshold from above.

In BAR Gossip [24], the authors present and prove the first P2P gossip-based data streaming application designed for the BAR model. The BAR model allows for Byzantine, altruistic and rational (selfish) nodes. Their protocol ensures predictable throughput even if some of the nodes are Byzantine and the rest are selfish. To overcome the randomness of gossip protocols, the authors suggest using verifiable pseudo-randomness to select peers. The authors assume an altruistic source that streams live content to a pool of clients. The Balanced Exchange protocol is used by clients to trade updates one-for-one. The Optimistic Push protocol, in which updates are sent without expecting anything in return, is used as a safety net. One of the strongest points of BAR Gossip is using cryptographic primitives as a means to prove misbehavior.

FlightPath [23] is a highly reliable gossip-based P2P streaming application based on

BAR Gossip that supports a dynamic set of peers using the BAR model. By using the approximate ϵ-Nash Equilibrium [10], the authors are able to prove cooperation while allowing for a bounded imbalance between peers, load balancing, and erasure codes. In an ϵ-Nash Equilibrium, a rational player deviates if and only if it expects to benefit by more than a factor of ϵ.

2.2 MANET Routing

A considerable number of papers dealing with selfish behavior in MANETs have been recently published [1, 2, 3, 5, 6, 7, 8, 12, 18, 26, 28, 29, 34, 37, 39, 40, 41, 42]. Most of these protocols use Dynamic Source Routing (DSR) [20] as the underlying routing protocol. DSR is then limited to only select routes that do not include non-cooperative nodes.

One approach to detect misbehavior is using virtual-currency (or credit) to encourage cooperation. All protocols currently using this scheme either require tamper-proof hardware [2, 6, 7, 8, 34] or a centralized credit manager [34, 39, 42]. In [7], for example, a source stores nuglets (virtual currency) in data packets before sending them as a payment to intermediate nodes - requiring tamper-proof hardware. SPRITE [42] uses an authorized centralized Credit Clearance Service to manage credit, but does not require tamper-proof hardware.

An alternate approach to dealing with selfishness is using first-hand neighbor reputations based on statistics gathered by each node to decide whether a given neighbor is cooperative or not [3, 5, 18, 26, 28]. To gather this information, a watchdog [26] is used to monitor packets sent by neighboring nodes by means of eavesdropping in promiscuous mode. The CONFIDANT protocol [5] adds second hand positive and negative reputation gathered from neighboring nodes so as to learn from their experience as well. Second hand reputation is reputation as reported by (perhaps untrusted) neighbors. Positive reputation states that a given node is cooperative, whereas negative reputation states the opposite. CORE [28] notes the possibility of cheating using false reports in CONFIDANT, and therefore solely uses positive reports. OCEAN [3] and later LARS [18] avoid second-hand reputation, and use only direct first-hand reputation to avoid false accusations.

A game-theoretic approach has been used in previous work to provably enforce cooperation [1, 14, 21, 23, 24, 29, 37, 40, 41]. Michiardi and Molva [29] provide an evaluation of the CORE [28] protocol using both a cooperative game approach and a non-cooperative game approach. Altman et al [1] use the framework of non-cooperative game theory to provide incentives for cooperation by means of a punishment scheme which is less aggressive than TFT. Félegyházi et al [14] propose a model based on game theory and graph theory to investigate equilibrium conditions of different packet forwarding strategies when taking the network topology into account. All works that prove cooperation consider only pairwise interactions. That is, one node punishes another only for selfish behavior that the latter has exhibited towards the former.

By and large, previous work has not taken message loss into account. The only exception we are familiar with is the MANET routing scheme due to Yu and Liu [40, 41]. In [40], the authors prove that a node's best strategy is not to forward more packets than its opponent. Unlike previous work, the authors assume a noisy environment in which packet loss is modeled as an i.i.d. Bernoulli random process. However, in that paper, the authors assume perfect monitoring. Imperfect monitoring can be taken advantage of by dropping packets which may not be detected by the monitor. This assumption has been removed in a follow-up paper [41], where imperfect-monitoring is assumed.

In [41], Yu and Liu use a strategy similar to TFT between every two peers in the network. In this strategy, a node agrees to forward packets on behalf of another node only if the latter has previously forwarded enough packets for the former. A balance counter is used to count the balance of each node, i.e., the number of packets forwarded for the peer minus the number of packets forwarded by the peer. To decide whether a node should be considered selfish, under imperfect monitoring, [41] applies the Neyman-Pearson hypothesis testing theory [33] to find the threshold given an acceptable false alarm probability.

2.3 Comparison with LTSM

Each of the previously suggested solutions is built for a specific application. Furthermore, each current MANET routing solution deals with one specific routing protocol, usually

source routing (e.g., DSR [20]), and leave other protocols, with more favorable properties in large dynamic networks (e.g., LANMAR [31], GLS [25], and Octopus [27]), unresolved. In contrast, the solution we present here is general, and suitable for a wide range of P2P applications.

There are several important differences between our work and the work by Yu and Liu [41] described above. Whereas Yu and Liu focus on stimulating cooperation in a MANET source routing (DSR) protocol, we provide a general abstraction for P2P systems. Furthermore, Yu and Liu focus solely on packet forwarding, while LTSM allows for monitoring additional message types, such as route discovery, location queries and replies, and keep-alive messages. Third, their work employs a tit-for-tat (TFT) strategy between each pair of nodes in the network. In this strategy, a node agrees to forward packets on behalf of another node only if the latter has previously forwarded enough packets for the former. We, on the other hand, do not assume any specific strategy. Fourth, their analysis only shows how to set the cooperation threshold for a given desired false positive probability. They do not analyze how the false positive probability should be chosen so as to ensure cooperation at a minimal cost for cooperative nodes as we do. Moreover, their punishment scheme is quite draconic as there is no way for selfish nodes to be added back, which suggests that the false positive probability should be chosen to be very small. In contrast, we allow nodes suspected of non-cooperation to be added back by paying a fine.

Chapter 3

Monitoring Service Definition

We consider a P2P system in which the participating nodes are selfish and rational, i.e., each node wishes to participate in the protocol while choosing a strategy that minimizes its cost. A strategy consists of deciding which packets to transmit, out of the packets required by the P2P protocol. We say that a node is *cooperative* if it sends all the packets required by the protocol, and *non-cooperative* otherwise. For simplicity, we assume that the cost of sending all packets is the same.

We model the system as a *non-cooperative* game, in which the participating nodes are the players. A non-cooperative game is a game in which any cooperation between the players (nodes) is self-enforcing, i.e., contracts are not enforced by third parties. Our goal is to provide a monitoring service for P2P applications that makes the full cooperation of rational nodes a Nash Equilibrium, while minimizing the expected cost of cooperative nodes, and taking message loss into account. A Nash Equilibrium is a set of strategies such that each node's strategy is an optimal response to the other nodes' strategies [15]. I.e., a strategy profile is a Nash Equilibrium if no unilateral strategy deviation by any single node is profitable for that node.

Nodes may join and leave the system dynamically. Nodes can be removed from the system, e.g., due to misbehavior, and may be allowed back after paying an application-defined *fine*, specified in terms of the packet sending cost. For example, *fine*=7 means that a suspected node has to send seven penalty packets in order to be allowed back into the system. As free admission can be abused in systems subject to Sybil attacks [13], some

sort of payment is required in order to join such a system. One can set the cost of joining to be the same as the fine.

We classify messages in a P2P protocol into two categories. The first category consists of messages that are generated by nodes at a predetermined rate, one per a given *time unit*. Examples include keep-alive messages and data packets in a Constant Bitrate (CBR) stream. A CBR stream is possible in a lossy environment if the protocol requires each node to compensate for lost packets by sending empty packets instead, as done, for example, in [21, 24]. The second category includes request-based messages, whose sending is triggered by the receipt of other messages at unpredictable times, e.g., forwarding data in a routing protocol or sending a piece of content in response to a request in a content distribution protocol.

We consider a lossy underlying network in which packet loss is independent and identically distributed (i.i.d.) with probability p. Note that whenever a node detects the loss of its own packet, it may simply retransmit the packet to avoid being suspected of selfish behavior. We therefore restrict our attention to the case that a node cannot detect whether its own message has been lost.

The interface of our monitoring service is as follows: *start_monitor(N)* is invoked by the application when either a new node, N, is discovered, or when an allegedly non-cooperative node rejoins the system after paying a fine; *missed_message(N)* is called when a message the P2P application is expecting from N does not arrive in a timely fashion, and *detected_message(N)* is called when such an expected message is detected on time. An expected message is a message that should be sent according to the P2P application's protocol. Lastly, the *is_selfish(N)* predicate indicates whether node N is allegedly non-cooperative.

Chapter 4

LTSM Algorithm

The LTSM service running at every node keeps an activity record for every monitored node. A node is deemed non-cooperative if it has sent less than τ out of W expected messages, where W is a parameter window size, and τ is the detection threshold. The windows in the protocol do not overlap – all counters are reset after each window of size W. W and τ are measured in number of messages. The relations between W, τ, and the fine are analyzed in Chapter 4.1.

Algorithm 4.1 LTSM(W, τ)

start_monitor(N)

1: $X[N] \leftarrow 0$ {expected packets}
2: $Y[N] \leftarrow 0$ {detected packets}
3: *is_selfish[N]* $\leftarrow$ **false**

is_selfish(N)

1: **return** *is_selfish[N]*

detected_message(N)

1: **if** *is_selfish[N]*=**false then**
2: $Y[N] \leftarrow Y[N] + 1$
3: *advance_window(N)*

missed_message(N)

1: **if** *is_selfish[N]*=**false then**
2: *advance_window(N)*

advance_window(N)

1: $X[N] \leftarrow X[N] + 1$
2: **if** $X[N]$=W **then**
3: **if** $Y[N] \leq \tau$ **then**
4: *is_selfish[N]* $\leftarrow$ **true**
5: **else** {start a new window}
6: $X[N] \leftarrow 0$
7: $Y[N] \leftarrow 0$

The LTSM protocol is depicted in Algorithm 4.1. The activity record of each mon-

itored node, N, consists of two counters and a boolean, *is_selfish[N]*, which indicates whether the node is allegedly non-cooperative. The first counter, $X[N]$, counts the number of packets expected by the P2P protocol to arrive in the current window. The second counter, $Y[N]$, counts the number of packets detected in the current window. The *start_monitor(N)* method resets all these counters to zero and sets *is_selfish[N]* to *false*. The *is_selfish(N)* method simply returns the boolean *is_selfish[N]*. Recall that the *missed_message(N)* method is called by the P2P application when a message expected by the P2P protocol does not arrive on time. Hence, this method advances the window by increasing the expected messages counter, $X[N]$. A decision is made whether node N should be deemed non-cooperative at the end of the window, i.e., when the $X[N]$ counter reaches W. If N is declared cooperative (not selfish), then all counters are reset, and a new window begins. Lastly, recall that the *detected_message(N)* method is called when an expected message is detected. This method increases the detected messages counter, $Y[N]$, and then advances the window as in *missed_message(N)*. Notice that no counters advance in case the node is alleged non-cooperative.

Algorithm 4.2 LTSM generic usage examples

on request to serve(N)

1: **if** *is_selfish[N]*=**false then**
2: serve request

CBR(N)

every time unit do

1: **if** received packet **then**
2: *detected_message(N)*
3: **else**
4: *missed_message(N)*

on receive fine from N

1: *start_monitor(N)*

Request-based(N)

on detect request do

1: wait timeout
2: **if** received response packet **then**
3: *detected_message(N)*
4: **else**
5: *missed_message(N)*

Algorithm 4.2 shows two generic usage examples for LTSM, one of a CBR stream, and one of request-based traffic. We leave out the initializations, and simply assume that *start_monitor(N)* has been called for each node N. In a CBR stream, a timer is used to check whether a packet arrives at every (predetermined) time unit; *detected_message(N)* is called if such a packet arrives, and *missed_message(N)* is called otherwise. Similarly,

for request-based traffic, a timeout is used to determine whether a given request yields a response. An allegedly non-cooperative node is denied service until it pays a fine. It is important to note that LTSM's parameters (*W*, *fine*, and τ) differ for CBR and request-based traffic monitoring (see Chapter 4.1). More detailed usage examples are provided in Chapter 5.

In Chapter 4.1, we show that using a large W lowers the per-packet expected cost, *Ecost(W)/W*. On the other hand, with a high W, it takes LTSM a long time to detect a non-cooperative node, as a node's misbehavior status is computed and output by LTSM once per window. We mitigate this issue by using a *sliding window*. A *Sliding Window LTSM* works very similarly to LTSM.

A cyclic bit vector of size W, $Y_b[N]$, is used instead of the $Y[N]$ counter to keep a record of the number of packets detected, out of the last W expected packets. All bits in $Y_b[N]$ are initially set. The *missed_message(N)* and *detected_message(N)* methods advance the window by setting $X[N] = (X[N] + 1) \bmod W$, and *clearing* or *setting* bit $X[N]$ in $Y_b[N]$, respectively. A node N is deemed non-cooperative if the number of bits set in $Y_b[N]$ is less or equal to τ. For clarity of the exposition, we analyze the non-sliding window version of LTSM in the following chapter. The results trivially apply to Sliding Window LTSM.

4.1 Analysis

Our goal is to understand the relations between *W*, τ, the fine, the loss rate *p*, and the expected costs of cooperative and non-cooperative nodes. These relations help us determine the best parameter choices for the protocol, so that full cooperation and fully following the protocol is a Nash Equilibrium. We define $q = 1 - p$. We first analyze constant-rate traffic (CBR) in Chapter 4.2, then provide several graphs with numerical examples in Chapter 4.3, and finally extend the analysis to request-based traffic (RB) in Chapter 4.4.

4.2 Analysis for Constant Rate Traffic

Due to the CBR nature of the traffic, a single packet is expected to arrive per time unit. As packet loss cannot be detected by the sending node itself (See Chapter 3), the decision whether to send a given packet is independent of previous events. We therefore assume that each node decides in advance on the number of packets it sends in a given window.

Consider a node I monitoring another node F. We use the following notations:

$$y(X) \triangleq \text{the number of packets received at I out of X packets sent by F}$$
$$D(X,\tau) \triangleq P(y(X) \leq \tau), \text{ detection probability}$$
$$\epsilon(\tau) \triangleq D(W,\tau) = P(y(W) \leq \tau), \text{ false positive probability}$$

Let $y(X)$ be a random variable representing the number of packets that are received at I in a given window, out of X packets sent by F. Since loss is i.i.d. with probability $p = 1 - q$, $y(X)$ is a binomial random variable, *y(X)∼Binomial(X,q)* [38]. F is detected as faulty unless more than τ packets arrive. The detection probability is therefore equal to the binomial cumulative distribution function at τ,

$$D(X,\tau) = P(y(X) \leq \tau) = \begin{cases} I_p(X-\tau, \tau+1) & \text{if } X > \tau, \\ 1 & \text{otherwise,} \end{cases} \tag{4.1}$$

where I is the *regularized incomplete beta function* [38].

We now turn to compute the expected cost of sending X packets in a window. Recall that the sender has to pay a fine to continue participating, in case the number of packets received, $y(X)$, is lower than the detection threshold, τ. Thus, the expected cost, $Ecost(X,\tau)$, when sending X packets is:

$$Ecost(X,\tau) = X + fine \times D(X,\tau). \tag{4.2}$$

Our goal is to find *fine*, *W*, and τ that encourage full cooperation, i.e., ensuring that for a given *W*, and all natural $1 \leq n \leq W$, $Ecost(W-n,\tau) > Ecost(W,\tau)$. That is, the expected cost of sending less than W messages is strictly higher than the expected

cost of sending W messages, assuming that the sender intends to continue participating in the protocol. The required relations among the parameters are captured by the following lemma.

Lemma 4.1. *For all natural n, $0 < n \leq W$: $Ecost(W-n,\tau) > Ecost(W,\tau)$ if and only if the following constraint holds:*

$$fine > \max\left(\frac{W}{1-\epsilon(\tau)}, \frac{1}{qP(y(W-1)=\tau)}\right).$$

We now prove several auxiliary claims, which are used to prove Lemma 4.1. The first claim shows that the fine needs to exceed a certain bound, captured by the following function:

Definition 1 (Lower Bound on Fine)**.** *For all natural n, $0 < n \leq W$:*

$$f(\tau,W,n) \triangleq \frac{n}{D(W-n,\tau)-\epsilon(\tau)}.$$

Claim 1. *For all natural n, $0 < n \leq W$: $Ecost(W-n,\tau) > Ecost(W,\tau)$ if and only if $fine > f(\tau,W,n)$.*

Proof. From (4.2) we have: $Ecost(W,\tau) = W + fine \times \epsilon(\tau)$, and $Ecost(W-n,\tau)] = W - n + fine \times D(W-n,\tau)$. Solving $Ecost(W-n,\tau) > Ecost(W,\tau)$ and isolating $fine$ we get the desired result. □

Therefore, we are looking for a fine so that $fine > f(\tau,W,n)$ for all natural n, $0 < n \leq W$. We first show that if a rational node sends less than τ messages, then it sends no messages at all.

Claim 2. *For all natural n, $W-\tau < n \leq W$: $Ecost(W-n,\tau) > Ecost(W,\tau)$ if and only if $fine > \frac{W}{1-\epsilon(\tau)}$.*

Proof. The expected cost for all $X \in [0,\tau)$, which is $X + fine$, is minimized at $X = 0$. Therefore, the proof follows from Claim 1 with $n = W$. □

Having resolved the case that less than τ packets are sent, we now restrict our attention to the case that τ to $W-1$ packets are sent, i.e., n is in $(0, W-\tau]$. We next simplify the expression for $f(\tau,W,n)$ in order to find a constraint that enforces full cooperation.

Claim 3. *For all natural* m, $\tau \leq m < W$: $D(m,\tau) = D(m+1,\tau) + qP(y(m) = \tau)$.

Proof. We use the following transformation of the regularized incomplete beta function [38] using the Gamma function Γ:

$$I_z(a+1,b) = I_z(a,b) - \frac{\Gamma(a+b)}{\Gamma(a+1)\Gamma(b)}(1-z)^b z^a.$$

From (4.1), we have $D(m+1,\tau) = I_p(m-\tau+1,\tau+1)$. Therefore,

$$\begin{aligned} D(m+1,\tau) &= I_p(m-\tau,\tau+1) - \frac{\Gamma(m+1)}{\Gamma(m-\tau+1)\Gamma(\tau+1)} q^{\tau+1} p^{m-\tau} \\ &= D(m,\tau) - q\frac{m!}{(m-\tau)!\tau!} q^\tau p^{m-\tau} = D(m,\tau) - qP(y(m) = \tau). \end{aligned}$$ □

The following claim shows that $f(\tau, W, n)$ (Definition 1) is one over the average of $P(y(W-k) = \tau)$ over $k \in [1,n]$, multiplied by q. Recall that $W - n$ here is the number of packets sent.

Claim 4. *For all natural* $0 < n \leq W - \tau$:

$$f(\tau, W, n) = \frac{n}{q\sum_{k=1}^{n} P(y(W-k) = \tau)} \ .$$

Proof. We start by iteratively applying Claim 3 on $D(W-n,\tau)$: $D(W-n,\tau) = D(W-n+1,\tau) + qP(y(W-n) = \tau) = D(W,\tau) + q\sum_{k=1}^{n} P(y(W-k) = \tau)$. Recall that $\epsilon(\tau) = D(W,\tau)$. The result follows by substituting $D(W-n,\tau)$ into the definition of $f(\tau, W, n)$. □

We thus have an expression for $f(\tau, W, n)$, which is a lower bound for the fine (see Claim 1). To ensure full cooperation the bound has to hold for all n. We therefore seek the maximum value of $f(\tau, W, n)$. We start with an auxiliary claim.

Claim 5. *Let* $a = \{a_k\}_{k=1}^m$ *be a series that increases for* $k < A$ *and decreases for* $k > A$ *for some real number A. Let* $z = \{z_n = \frac{1}{n}\sum_{k=1}^{n} a_k\}_{n=1}^m$ *be the series of partial averages of* a. *Then the minimum of the series* $\{z_n\}_{n=1}^m$ *occurs at* $n = 1$ *or* $n = m$. *That is,* $\min\{z_n\}_{n=1}^m = \min(z_1, z_m)$.

Proof. Observe that for all natural n, $1 \leq n < m$, $z_{n+1} > z_n$ if and only if $a_{n+1} > z_n$. Since a increases for $k < A$, z also increases for $n < A$. Although a decreases for $k > A$, z may continue increasing for some $n > A$, as long as $a_{n+1} > z_n$. If there is no n, $1 \leq n < m$, such that $a_{n+1} \leq z_n$, then z is always increasing and $\min\{z_n\}_{n=1}^{m} = z_1$. Otherwise, let j be the smallest index $1 \leq j < m$ so that $a_{j+1} \leq z_j$. Then z is increasing for $k < j$. It is easy to see, by induction on $k \geq j$, that z is decreasing for $k \geq j$. Thus, $\min\{z_n\}_{n=1}^{m} = min(z_1, z_m)$. □

Claim 6. *The maximal value of $f(\tau, W, n)$ for $0 < n \leq W - \tau$ occurs at $n = 1$ or $n = W-\tau$. That is,* $\max\{f(\tau, W, n)|0 < n \leq W-\tau\} = \max\{f(\tau, W, n)|n = 1, W-\tau\}$.

Proof. Since y(X) is a binomial random variable, for all natural k, $0 < k < W - \tau$:

$$\frac{P(y(W-k)=\tau)}{P(y(W-(k+1))=\tau)} = \frac{\frac{(W-k)!}{\tau!(W-k-\tau)!} \times q^{\tau} p^{W-k-\tau}}{\frac{(W-k-1)!}{\tau!(W-k-1-\tau)!} \times q^{\tau} p^{W-k-1-\tau}} = \frac{W-k}{W-k-\tau} \times p.$$

Hence, $P(y(W-k) = \tau)$ is increasing for $k < W - \tau/q$, and decreasing for $k > W - \tau/q$. By Claim 5, the average, $\frac{1}{n}\sum_{k=1}^{n} P(y(W-k) = \tau)$, is minimized at one of the extremities ($n = 1$, $n = W - \tau$). Hence, the maximum of $f(\tau, W, n)$, which is one over the above average divided by q (see Claim 4), also occurs at either $n = 1$ or $n = W - \tau$. □

We are now ready to prove Lemma 4.1, which is restated below.

Lemma 4.1 (restated). *For all natural n, $0 < n \leq W$: $Ecost(W-n, \tau) > Ecost(W, \tau)$ if and only if the following constraint holds:*

$$fine > \max\left(\frac{W}{1-\epsilon(\tau)}, \frac{1}{qP(y(W-1)=\tau)}\right).$$

Proof. By Claim 2, for all natural n such that $W - \tau < n \leq W$: $Ecost(W-n, \tau) > Ecost(W, \tau)$ if and only if $fine > \frac{W}{1-\epsilon(\tau)}$. For $0 < n \leq W - \tau$, by Definition 1 and

Claim 4:

$$f(\tau, W, n = W - \tau) = \frac{W - \tau}{P(y(\tau) \leq \tau) - \epsilon(\tau)} = \frac{W - \tau}{1 - \epsilon(\tau)},$$
$$f(\tau, W, n = 1) = \frac{1}{qP(y(W-1) = \tau)}.$$

By Claim 6, the constraint for $0 < n \leq W - \tau$ is the the maximum of the above two values. Hence, the constraint for $0 < n \leq W$ is given by:

$$\max\left(\frac{W - \tau}{1 - \epsilon(\tau)}, \frac{1}{qP(y(W-1) = \tau)}, \frac{W}{1 - \epsilon(\tau)}\right). \qquad \square$$

Corollary 4.1. *Lemma 4.1 provides a constraint that must be upheld in order to make full cooperation a Nash Equilibrium.*

We now turn to minimizing the expected cost of a cooperating node under this constraint. This is a discrete optimization problem.

To solve this problem, we convert it to a continuous problem by using an approximation for large values of W. A common rule-of-thumb is that if both Wp and Wq are greater than 5 [22], then the binomial distribution can be approximated by the normal distribution. E.g., for a loss probability of $p = 0.1$, $W = 50$ should suffice.

Somewhat surprisingly, the following lemma (Lemma 4.2) shows that the expected cost for a cooperative node is minimized at $\tau = 0$, i.e., a single packet arriving at the destination is sufficient for the node to be considered cooperative. This, however, is obtained at the expense of a very high $fine$ value.

Lemma 4.2. *For a large enough W, under the constraint of Lemma 4.1, the expected cost for a cooperative node is minimized at $\tau = 0$. The minimal expected cost is $W + p/q$, and the fine has to satisfy $fine > 1/(qp^{W-1})$.*

In order to prove Lemma 4.2, we first prove three auxiliary claims. The following function, f_{max}, captures the maximum value of f for a given false positive probability δ:

$$f_{max}(\delta, W) \triangleq \max_{1 \leq n \leq W} f(\epsilon^{-1}(\delta), W, n).$$

We start by looking for an approximation for $\epsilon^{-1}(\delta)$ (for a large W), using the error function, erf [38].

Claim 7. *Let $\epsilon(\tau) = \delta$. Then for a large W, $\tau \approx qW - 0.5 - \sqrt{2pqW}(\mathrm{erf}^{-1}(1-2\delta))$.*

Proof. For a large W, we can use the de Moivre-Laplace Theorem [38]:

$$\begin{aligned}\delta = P(y(W) > \tau) &= \Phi\left(\frac{\tau + 0.5 - \mu}{\sigma}\right) + O\left(\frac{1}{\sqrt{W}}\right) \\ &= \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{\tau + 0.5 - \mu}{\sqrt{2}\sigma}\right)\right] + O\left(\frac{1}{\sqrt{W}}\right),\end{aligned}$$

where Φ is the standard normal cumulative distribution function, $\mu = Wq$, and $\sigma^2 = Wpq$. Hence, $\tau = \epsilon^{-1}(\delta) \approx \mu - 0.5 - \sigma\sqrt{2}\mathrm{erf}^{-1}(1-2\delta)$. □

Next, we approximate $P(y(W-1) = \tau)$, which appears in the constraint of Lemma 4.1.

Claim 8. *For a large W, if $\epsilon(\tau) = \delta$, then*
$P(y(W-1) = \tau) \approx (1/\sqrt{2\pi(W-1)pq}) \times \exp(-(\mathrm{erf}^{-1}(1-2\delta))^2)$.

Proof. For a large W, using the de Moivre-Laplace Theorem with $\mu = (W-1)q$, and $\sigma^2 = (W-1)pq$:

$$P(y(W-1) = \tau) = \binom{W-1}{\tau} q^{\tau} p^{(W-1-\tau)} \approx \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\tau-\mu)^2}{2\sigma^2}\right).$$

Using the approximation of τ from Claim 7, we get:

$$\begin{aligned}P(y(W-1) = \tau) &\approx \frac{1}{\sqrt{2\pi(W-1)pq}} \exp\left[-\left[\frac{-\frac{1}{2} - \sqrt{2Wpq}\mathrm{erf}^{-1}(1-2\delta) + q}{\sqrt{2(W-1)pq}}\right]^2\right] \\ &\approx \frac{1}{\sqrt{2\pi(W-1)pq}} \exp[-(\mathrm{erf}^{-1}(1-2\delta))^2].\end{aligned}$$ □

Claim 9. *$\epsilon(0) = p^W$ and $f_{max}(\epsilon(0), W) = 1/(p^{W-1}q)$.*

Proof. When $\tau = 0$, a false suspicion occurs if and only if all W messages are lost,

therefore $\epsilon(0) = p^W$. Let $\delta = \epsilon(\tau)$. By Lemma 4.1,

$$f_{max}(\delta, W) = \max\left(\frac{W}{1-\delta}, \frac{1}{qP(y(W-1) = \epsilon^{-1}(\delta))}\right) = \max\left(\frac{W}{1-p^W}, \frac{1}{qp^{W-1}}\right).$$

It remains to show that $\frac{W}{1-p^W} \leq \frac{1}{qp^{W-1}}$, i.e., that $W \leq \frac{1-p^W}{qp^{W-1}}$. We prove this by induction. For $W = 1$, both sides of the inequality are equal. Assume that the inequality holds for $W = k$, $k \in \mathbb{N}$. We have to prove that the inequality still holds for $W = k + 1$. To this end, we subtract the inequality for $W = k$ from the inequality for $W = k + 1$, and get:

$$1 = k + 1 - k \leq \frac{1 - p^{k+1}}{qp^k} - \frac{1 - p^k}{qp^{k-1}} = \frac{1 - p^{k+1} - p + p^{k+1}}{qp^k} = \frac{1}{p^k} \ ,$$

which holds for all $0 < p \leq 1$ and all k. □

We are now ready to prove Lemma 4.2, which is restated below.

Lemma 4.2 (restated). *For a large enough W, under the constraint of Lemma 4.1, the expected cost for a cooperative node is minimized at $\tau = 0$. The minimal expected cost is $W + p/q$, and the fine has to satisfy $fine > 1/(qp^{W-1})$.*

Proof. The expected cost for a cooperative node is $W + fine \times \epsilon(\tau)$. Let $\delta = \epsilon(\tau)$. We start by looking at the second term of the constraint in Lemma 4.1. Let

$$g(\delta, W) \triangleq \frac{1}{qP(y(W-1) = \epsilon^{-1}(\delta))}.$$

By Claim 8, for a large W, the asymptotic behavior of $g(\delta, W) \times \delta$ (for constant *W*, *p*, and *q*) is:

$$g(\delta, W) \times \delta \propto \frac{\delta}{\exp(-(\mathrm{erf}^{-1}(1-2\delta))^2)},$$

which is increasing in δ. Hence, the minimum of $g(\delta, W) \times \delta$ is obtained at the minimal $\delta = p^W$, where $g(\delta = p^W, W) \times \delta = \frac{\delta}{p^{W-1}q} = \frac{p}{q}$.

Now look at the first term of the constraint in Lemma 4.1.
Let $h(\delta, W) \triangleq W/(1-\delta)$. Notice that $h(\delta, W) \times \delta = (\delta \times W)/(1-\delta)$ is also an increasing function in δ, with a minimum at $\delta = p^W$.

By Claim 9, we get:

$$fine > \max\{g(p^W, W), h(p^W, W)\} = f_{max}(p^W, W) = \frac{1}{p^{W-1}q} \ . \qquad \square$$

Although Lemma 4.2 is stated (and proven) only for large values of *W*, we have empirically observed that the result actually holds for all values of W.

Though Lemma 4.2 identifies the optimal fine in terms of overall cost, it calls for very high fines. For example, with $p = 0.1$ and $W = 50$, the fine, according to Lemma 4.2, should be higher than 10^{49}. In Chapter 4.3, we illustrate some numerical examples, and show that W has a similar influence on the cost: the higher the fine and the window size are, the lower the expected cost for cooperative nodes is. On the other hand, with a high W, it takes LTSM a long time to detect a non-cooperative node. Similarly, a high fine can be detrimental for performance, especially in systems that are susceptible to Sybil attacks, where a high fine would entail a high join cost. Thus, there is a trade-off involved in setting these parameters.

A typical P2P application would therefore optimize its cost under additional constraints on the highest acceptable fine and W. Nevertheless, Lemma 4.1 requires the fine to be greater than $W/(1 - \epsilon(\tau))$, which is at least greater than W. Thus, not all values of $fine$ and W are feasible.

Let erf be the error function [38], and

$$\tau_{min} \triangleq qW - 0.5 - \sqrt{2pqW}\sqrt{-\ln\left(\frac{\sqrt{2\pi(W-1)pq}}{q \times fine}\right)}, \tag{4.3}$$

$$\delta_{min} \triangleq \frac{1 - \mathrm{erf}\left(\sqrt{-\ln\left(\frac{\sqrt{2\pi(W-1)pq}}{q \times fine}\right)}\right)}{2}. \tag{4.4}$$

The following lemma shows that setting the threshold value (τ) to τ_{min} defined above warrants full cooperation and yields a minimal expected cost, given pre-defined W and $fine$, and assuming that the constraint of Lemma 4.1 holds, which is captured using δ_{min} above.

Lemma 4.3. *Given $fine$ and W such that W is large, if $fine > W/(1 - \delta_{min})$, and*

$\delta_{min} < 0.5$*, then choosing* $\tau = \lfloor \tau_{min} + 1 \rfloor$ *warrants that full cooperation is a Nash Equilibrium and yields a minimal expected cost over all possible values of* τ*.*

Proof. Let $\delta = \epsilon(\tau)$. Recall that the expected cost for a cooperative node is equal to $W + \delta \times fine$. As W and $fine$ are given constants, the expected cost is minimized, while warranting full cooperation, by finding the smallest δ that satisfies the constraint of Lemma 4.1:

$$fine > f_{max}(\delta, W) = \max\left(\frac{W}{1-\delta}, \frac{1}{qP(y(W-1) = \epsilon^{-1}(\delta))}\right)$$

The lemma's preconditions guarantee the first constraint, which is satisfied for all $\delta < 1 - W/fine$. Using Claim 8, the second constraint becomes:

$$fine > \frac{\sqrt{2\pi(W-1)pq}}{q\exp(-(\mathrm{erf}^{-1}(1-2\delta))^2)},$$

which is decreasing in δ for $\delta < 0.5$. Thus, from the second constraint, with $\delta < 0.5$, we get:

$$\delta > \delta_{min} = \frac{1 - \mathrm{erf}\left(\sqrt{-\ln\left(\frac{\sqrt{2\pi(W-1)pq}}{q \times fine}\right)}\right)}{2}$$

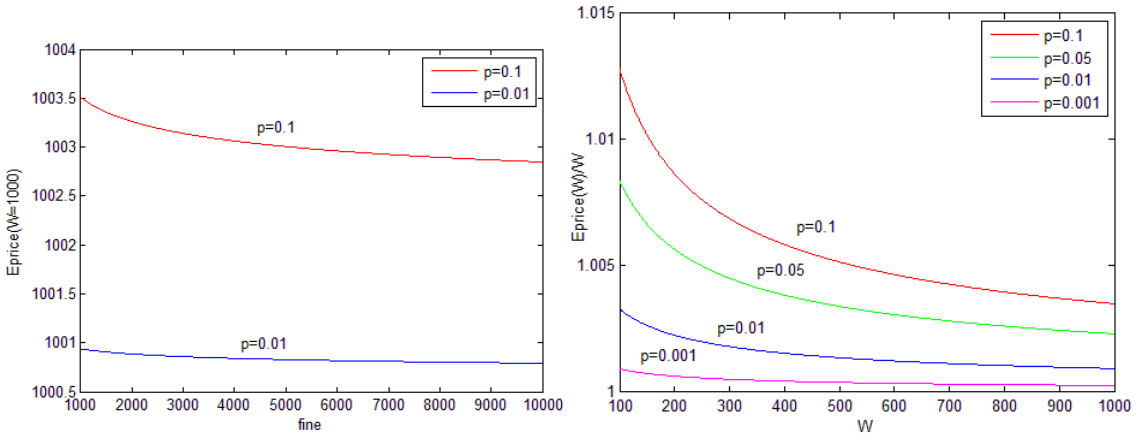
The result for τ_{min} is attained by substituting δ_{min} into the expression for τ given in Claim 7. □

Notice that feasible values of W and $fine$ can be found numerically using Lemma 4.3.

4.3 Numerical Examples

For fixed $fine$ and W, we compute the optimal τ using Lemma 4.3. Figure 4.1a depicts the resulting expected cost for $W = 1000$ as a function of the fine for two loss probabilities ($p = 0.1$, $p = 0.01$). We see that the higher the fine is, the lower the expected cost is (as predicted by Lemma 4.2). However, choosing a fine significantly lower than the optimal according to Lemma 4.2 is not too costly, as the expected cost decreases rather slowly as $fine$ increases. The reasoning behind this behavior is that for a higher $fine$,

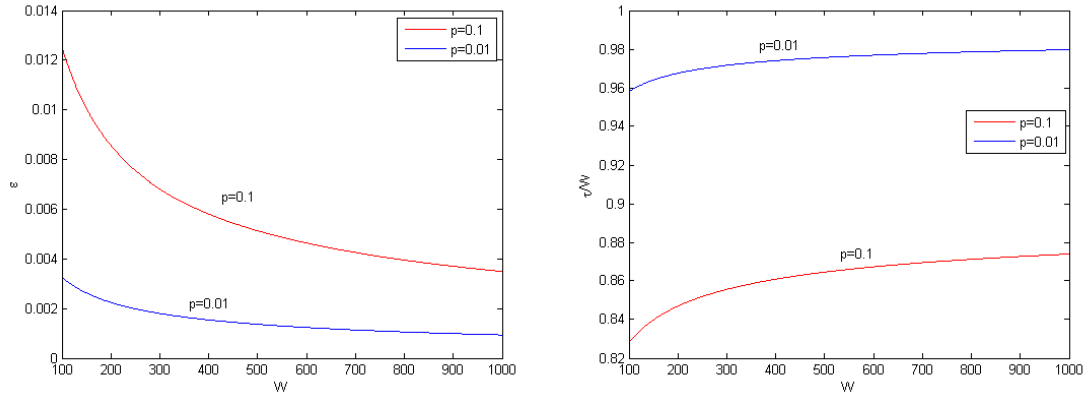
a lower τ is sufficient to discourage selfish behavior. A lower τ results in a lower false positive probability ,$\epsilon(\tau)$, which decreases super-linearly in $fine$.



(a) Expected cost for W=1000

(b) Expected cost divided by W for the minimal possible fine

Figure 4.1: The impact of W, fine, and p on the expected cost.



(a) False positive probability

(b) Detection threshold, τ, divided by W

Figure 4.2: The effect of W.

Figure 4.1b depicts the normalized expected cost, i.e., $Ecost(W, \tau)$ divided by W, for various window sizes, using the smallest fine satisfying the constraint of Lemma 4.1. We see that increasing W significantly reduces the normalized expected cost. Notice that the value for δ_{min} as defined in (4.4) grows sub-linearly in W. Recall that the expected cost for a cooperative node is equal to $W + \delta \times fine$. Thus, if only $fine$ is constrained by

the application (and not W), then the largest W that is feasible according to Lemma 4.3 minimizes the normalized expected cost, $Ecost(W, \tau)/W$.

The false positive probability decreases rapidly as W increases, as we can see in Figure 4.2a. We use the smallest fine satisfying the constraint of Lemma 4.1 for each W. The reasoning behind this behavior is that τ/W increases in W, as can be seen from (4.3), which lowers the false positive probability. Figure 4.2b shows that the detection threshold, τ, becomes closer and closer to W as W increases, when using the smallest fine as described above.

Figure 4.3 depicts the smallest feasible fine, used in the calculations of Figure 4.2 above, minus W, for various window sizes. The smallest feasible fine that makes full cooperation and fully following the protocol a Nash Equilibrium for a given W is found numerically using Lemma 4.3. Recall that the smallest feasible fine for a given W is a fine that satisfies the constraint of Lemma 4.1. One can see that such a fine is only slightly larger than W. The reasoning behind this behavior is that the false positive probability, $\epsilon(\tau)$, decreases super-linearly in *fine*. Recall that according to Lemma 4.1, *fine* is bounded from below by $W/(1 - \epsilon(\tau))$.

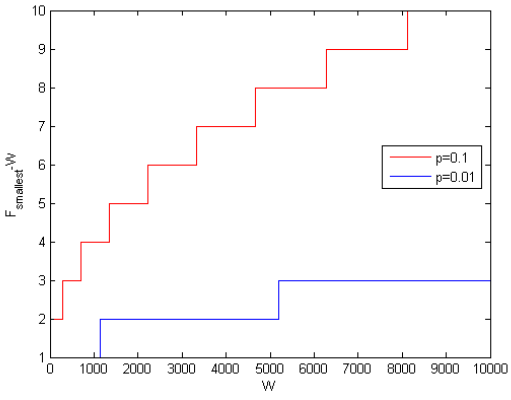


Figure 4.3: *Smallest fine satisfying the constraint of Lemma 1 minus W*

4.4 Analysis for Request-Based Traffic

We now adapt our previous discussion to request-based traffic (such as data forwarding in a routing protocol). Let I be an inspecting node at which we measure the counters, X and Y, and let F be the inspected node. A request message that should trigger request-based sending may be issued by I itself, or, in some applications, by some other node S. In any case, I must be able to detect both the request and the response messages. We assume a single response message per request message. Upon detecting a request message, node I may falsely accuse F of a packet loss in two cases:

1. F does not receive the request message; or
2. F receives the request message, but I does not receive F's response.

Hence, the probability that I falsely accuses F is $\tilde{p} = p + qp = 1 - q^2$, instead of p as in the previous chapter.

Let W be the window size (in messages). As in the previous chapter, I decides whether a node is behaving selfishly or not at the end of each such window. The threshold, τ, and the fine value that ensure full cooperation are obtained the same way as in the previous chapter, by using $\tilde{p}$ instead of p.

In case more than one response is expected for a given request, the requesting node should use a CBR monitor in conjunction with a request-based monitor. The requesting node should begin CBR monitoring upon hearing the first response. If no response is obtained within a timeout, the requester should re-send the request. Eventually, either a response arrives and CBR monitoring begins, or the request-based monitor suspects the node.

Note that in some applications there may be messages that F is asked to send by S which are unseen by I. Such messages may safely be dropped by F without fearing detection by I. However, in most cases, F has no way of knowing whether I has heard a message or not. In some cases, such as a MANET in which node's locations and radio ranges are known, F may drop packets knowing it won't be detected by I. On the other hand, it may be detected by other nodes.

Chapter 5

Applications

In this chapter we show two examples of how LTSM can help make P2P algorithms immune to rational selfish nodes, namely, multicast (Chapter 5.1) and MANET routing (Chapter 5.2).

5.1 Multicast

We first focus on tree-based multicast (either wired or wireless), which is the most common multicast architecture. We follow the common practice of assuming that the multicast source node is altruistic and trusted by all nodes [21, 24, 36]. The multicast protocol disseminates messages over an overlay tree, which spans all participating nodes. One way to build and maintain the multicast tree is using a trusted entity, such as the multicast source [19, 32], which eliminates selfish behavior in the tree-building stage. A distributed tree building scheme which is immune to selfish behavior (such as in [30]) can also be used. The tree must be changed dynamically to account for topology changes, which are common, especially in MANETs. For the remainder of this section, we assume an external mechanism for building the multicast tree.

Algorithm 5.1 Monitoring Service for Tree-Based Multicast

Source

on_suspicion_report(P)

1: **if** reporter is not child of P **then**
2: **return** {ignore}
3: ask P to pay a fine
4: **if** not received a fine **then**
5: Ask *parent(P)* to Disconnect P
6: Reconnect P's sub-tree

Client Node

on_join_tree(*P*, *cur_packet*)

1: *start_monitor(P)*
2: $i \leftarrow$ *cur_packet*

on_receive_fine_request

1: send a fine to the source

on_timer($1/r$)

1: **if** packet i received from P **then**
2: *detected_message(P)*
3: forward packet to children
4: **else**
5: *missed_message(P)*
6: send dummy packet to children
7: $i \leftarrow i + 1$
8: **if** *is_selfish(P)* **then**
9: report P to source
10: *start_monitor(P)*

Algorithm 5.1 employs the interface defined in Chapter 3 to monitor and punish non-cooperative nodes. Let r be the multicast rate. For simplicity, we assume that the multicast application must receive exactly one packet every $1/r$ seconds and that the packets are numbered. A node receives the current packet number and its parent's name from the source when it joins a multicast tree. It then uses LTSM to monitor the traffic forwarded by its parent. A message miss is registered if no packet is received from the parent in a given interval. A message detection is registered otherwise. A node that misses a message that it has to send to its children, compensates for it by sending a dummy packet instead. We assume that the cost of sending dummy packets is the same as that of sending data packets, and hence a selfish rational node has no reason to send dummy packets when it does have data to send.

Once a parent P is suspected of selfish behavior by LTSM at its child C, the child reports its suspicion to the source node (which is altruistic), and optimistically restarts LTSM. The source then asks the suspected node to pay it a fine. In case the suspected node does not comply, the source asks the suspect's parent to disconnect the non-cooperative node, and the orphan sub-tree reconnects to the spanning tree, bypassing the removed

node.

Note that falsely accusing a parent is not beneficial for a child. A false suspicion merely causes the parent to pay a fine, increasing its cost, but with no benefit to the accusing child. Moreover, if the parent fails to pay the fine, the child's utility may decrease due to missed data or reduced performance during the tree reconstruction phase. On the other hand, reporting genuine suspicion is vital to ensure future service. Therefore, adherence to the protocol is a Nash Equilibrium. Observe that a parent does not have to behave selfishly towards all of its children. It may behave selfishly towards one or more of its children.

In a similar way, LTSM can be integrated into mesh-based multicast solutions. For example, EquiCast [21] uses similar threshold-based detection, and fines for readmission. Replacing their selfishness monitor with LTSM automatically makes EquiCast robust to packet loss.

5.2 MANET Routing

A common heuristical approach to detecting selfish behavior in a MANET is using reputations. Such reputations are collected by nodes through monitoring the behavior of other nodes. The watchdog mechanisms used in CONFIDANT [5] and later in CORE [28], OCEAN [3] and others, use promiscuous mode to eavesdrop on neighboring nodes and monitor packet forwarding and other network activity.

LTSM can easily be integrated into such heuristical reputation mechanisms, seamlessly adding resilience to lossy networks. In the watchdog mechanism, every node keeps track of packets that flow in and out of each of its radio range neighbors. This modus operandi is appropriate for LTSM's tracking of request-based messages. LTSM's *missed_message(N)* method can be called when a packet dropped by node N is detected, while *detected_message(N)* can be called when a packet is correctly forwarded by node N. LTSM can monitor additional request-based traffic, such as route discovery or location queries and replies, as well as constant rate traffic such as keep-alive messages.

LTSM can also be integrated into existing schemes that prove cooperation. Such existing schemes consider pairwise interactions [14, 37, 40, 41]. Recall that in such schemes,

one node punishes another only for selfish behavior that the latter has exhibited towards the former. Integration with LTSM can enhance them with minimal expected costs in a lossy network, and a redemption mechanism.

Chapter 6

Conclusions

P2P systems in commercial applications often operate over lossy networks and are bound to experience selfish behavior. We have defined a general service for monitoring and discouraging selfish behavior, which is applicable to a variety of P2P protocols. We have presented such a monitoring service, called LTSM, which is suitable for networks subject to message loss. We have mathematically analyzed LTSM, and shown how to tune its parameters so as to encourage full cooperation, while minimizing the cost for cooperating nodes. Finally, we have shown usage examples of our service for multicast and for MANET routing.

Bibliography

[1] E. Altman, A. A. Kherani, P. Michiardi, and R. Molva. Non-cooperative forwarding in ad-hoc networks. In *4th International IFIP-TC6 Networking Conf.*, volume 3462, pages 486–498, 2005.

[2] L. Anderegg and S. Eidenbenz. Ad hoc-vcg: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *MobiCom '03*, pages 245–259, 2003.

[3] S. Bansal and M. Baker. Observation-based cooperation enforcement in ad hoc networks. Technical Paper, Stanford University, 2003.

[4] A. Blanc, Y.-K. Liu, and A. Vahdat. Designing incentives for peer-to-peer routing. In *INFOCOM 2005*, pages 374–385, 2005.

[5] S. Buchegger and J.-Y. Le-Boudec. Performance analysis of the confidant protocol. In *MobiHoc '02*, pages 226–236, 2002.

[6] L. Buttyán and J.-P. Hubaux. Enforcing service availability in mobile ad-hoc wans. In *MobiHOC '00*, pages 87–96, 2000.

[7] L. Buttyán and J.-P. Hubaux. Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks. Technical report, EPFL, 2001.

[8] L. Buttyán and J.-P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *MONET*, 8(5):579–592, 2003.

[9] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: high-bandwidth multicast in cooperative environments. In *SOSP '03*, pages 298–313, 2003.

[10] S. Chien and A. Sinclair. Convergence to approximate nash equilibria in congestion games. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 169–178, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[11] B. Cohen. Incentives build robustness in bittorrent. In *1st Workshop on the Economics of Peer-to-Peer Systems*, 2003.

[12] J. Dong, K. E. Ackermann, B. Bavar, and C. Nita-Rotaru. Mitigating attacks against virtual coordinate based routing in wireless sensor networks. In *WiSec '08*, pages 89–99, 2008.

[13] J. R. Douceur. The sybil attack. In *IPTPS '01*, pages 251–260, 2002.

[14] M. Félegyházi, J.-P. Hubaux, and L. Buttyán. Nash equilibria of packet forwarding strategies in wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(5):463–476, 2006.

[15] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[16] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *INFOCOM 2005*, pages 2235–2245, 2005.

[17] A. Habib and J. Chuang. Service differentiated peer selection: An incentive mechanism for peer-to-peer media streaming. In *Internation Workshop on Quality of Service (IWQoS '04)*, pages 171–180, 2004.

[18] J. Hu. Cooperation in mobile ad hoc networks. Technical report, CS Department, Florida State University, 2005.

[19] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. James W. O'Toole. Overcast: reliable multicasting with on overlay network. In *OSDI'00*, 2000.

[20] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[21] I. Keidar, R. Melamed, and A. Orda. Equicast: scalable multicast with selfish users. In *PODC '06*, pages 63–71, 2006.

[22] L. M. Leemis and K. S. Trivedi. A comparison of approximate interval estimators for the bernoulli parameter. *The American Statistician*, 50:63–68, 1996.

[23] H. C. Li, A. Clement, M. Marchetti, M. Kapritsos, L. Robison, L. Alvisi, and M. Dahlin. FlightPath: Obedience vs choice in cooperative services. In *OSDI'08*, 2008.

[24] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. Bar gossip. In *OSDI '06*, pages 191–204, 2006.

[25] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *MobiCom '00*, pages 120–130, 2000.

[26] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *MobiCom '00*, pages 116–123, 2000.

[27] R. Melamed, I. Keidar, and Y. Barel. Octopus: A fault-tolerant and efficient ad-hoc routing protocol. In *SRDS'05*, pages 39–49, 2005.

[28] P. Michiardi and R. Molva. Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security*, pages 107–121, 2002.

[29] P. Michiardi and R. Molva. A game theoretical approach to evaluate cooperation enforcement mechanisms in mobile ad hoc networks. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 3–5, 2003.

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

[31] G. Pei, M. Gerla, and X. Hong. Lanmar: landmark routing for large scale wireless ad hoc networks with group mobility. In *MobiHoc '00*, pages 11–18, 2000.

[32] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: an application level multicast infrastructure. In *USITS'01*, 2001.

[33] H. V. Poor. *An introduction to signal detection and estimation (2nd ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1994.

[34] B. Raghavan and A. C. Snoeren. Priority forwarding in ad hoc networks with self-interested parties. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[35] P. Rodriguez, S.-M. Tan, and C. Gkantsidis. On the feasibility of commercial, legal p2p content distribution. *SIGCOMM Comput. Commun. Rev.*, 36(1):75–78, 2006.

[36] M. Sirivianos, J. H. Park, X. Yang, and S. Jarecki. Dandelion: Cooperative content distribution with robust incentives. In *USENIX*, pages 157–170, 2007.

[37] V. Srinivasan, P. Nuggehalli, C.-F. Chiasserini, and R. R. Rao. An analytical approach to the study of cooperation in wireless ad hoc networks. *IEEE Transactions on Wireless Communications*, 4(2):722–733, 2005.

[38] E. W. Weisstein. Mathworld–a wolfram web resource. wolfram research, inc. http://mathworld.wolfram.com, 1998-2007.

[39] Y. Yoo, S. Ahn, and D. P. Agrawal. A credit-payment scheme for packet forwarding fairness in mobile ad hoc networks. *IEEE ICC*, 5:3005–3009, 2005.

[40] W. Yu and K. J. R. Liu. Game theoretic analysis of cooperation stimulation and security in autonomous mobile ad hoc networks. *IEEE Trans. Mob. Comput.*, 6(5):507–521, 2007.

[41] W. Yu and K. J. R. Liu. Secure cooperation in autonomous mobile ad-hoc networks under noise and imperfect monitoring: A game-theoretic approach. *IEEE Transactions on Information Forensics and Security*, 3(2):317–330, 2008.

[42] S. Zhong, J. Chen, and Y. R. Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *INFOCOM*, 2003.

דיכוי אנוכיות ברשתות שיתופיות

עם איבודי הודעות

אלכסנדר פרידמן

דיכוי אנוכיות ברשתות שיתופיות עם איבודי הודעות

חיבור על עבודת גמר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים בהנדסת חשמל

אלכסנדר פרידמן

הוגש לסנט הטכניון – מכון טכנולוגי לישראל
שבט תשס"ט חיפה פברואר 2009

עבודת הגמר נעשתה בהנחיית פרופסור חבר עידית קידר

בפקולטה להנדסת חשמל

הבעות תודה

אני רוצה להודות למנחה שלי, פרופ"ח עידית קידר, על כך שהעשירה את הידע שלי, קראה אינספור טיוטות ותמיד הייתה פתוחה לשאלותי. סבלנותה האינסופית והנחיתה הצמודה הם אלו אשר הביאוני לסיים בהצלחה מחקר זה.

אני מודה לחברתי רינת על שגרמה לי לחייך לאורך כל תקופת לימודי.

תודה אחרונה ומיוחדת למשפחתי ולחברי על תמיכתם ועידודם לאורך השנים.

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי

תקציר

פרוטוקולי Peer-to-Peer משמשים מגוון רחב של יישומים, כגון ניתוב ברשתות אל-חוטיות ניידות (MANET), הפצת מידע בשידור חי ללא תשתית (P2P multicast) ורשתות שיתוף קבצים. רשתות התקשורת אשר מעליהן פועלים פרוטוקולים מסוג זה נוטות לסבול מאיבוד חבילות. לדוגמה, קצב איבוד חבילות גבוהה אופייני לרשתות אלחוטיות כמו MANETs . יתר על כן, מערכות הפצת מידע בשידור חי, אשר מפיצות קול או תמונה (סרט), משתמשות לרוב בפרוטוקולי תקשורת בלתי אמינים, כגון UDP, כיוון שאפליקציות אלו מוכנות לסבול איבוד מידע חלקי.

הצמתים החברים במערכת P2P מספקים את כל המשאבים הנדרשים לפעילות המערכת; כל צומת צריך לתרום זיכרון, כוח עיבוד, רוחב פס ואנרגיה. כיוון שרוב הצמתים ברשתות אל-חוטיות ניידות מופעלים על-ידי סוללות, אנרגיה היא משאב יקר במערכות מסוג זה. במערכות P2P מסחריות קיימת סבירות גבוהה מאוד לכך שצמתים יפגינו התנהגות אנוכית, אשר יכולה להתבטא בסטייה מהפרוטוקול לצורך קיצוץ עלויות. מסיבה זו חשוב שפרוטוקולים אלו ימשיכו לעבודה בצורה ובביצועים תקינים, גם כאשר צמתים משתמשים בגרסאות אנוכיות של הפרוטוקול.

מחקר רב הוקדש בשנים האחרונות להתמודדות עם התנהגות אנוכית במגוון מערכות P2P (כגון ניתוב ב-MANET, הפצת מידע ושיתוף קבצים). למרות זאת, אתגרים רבים נותרו ללא מענה. בעבודות קודמות, למשל, לא הציגו או השתמשו בדמיון הרב בין פרוטוקולי P2P מהסוגים הנ"ל, אלא התמקדו בפרוטוקולים ספציפיים ובסביבה ספציפית אחת. במחקרים אודות פרוטוקולי ניתוב ב-MANET, למשל, התמקדו בפרוטוקול ניתוב מסוים אחד, למרות הדמיון הרב שבין הפרוטוקולים. רובם המכריע של המחקרים בחרו בניתוב מקור דינמי (DSR) והותירו פרוטוקולים אחרים, בעלי תכונות טובות יותר ברשתות דינמיות גדולות (כגון LANMAR, GLS, Octopus ואחרים), ללא פתרון. אתגר נוסף אשר לא זכה להתייחסות במרבית העבודות הקודמות הינו סוגיית איבודי ההודעות ברשתות תקשורת. ניטור וזיהוי של התנהגות אנוכית קשה ומאתגר יותר משמעותית, כאשר על מנגנון הניטור להתמודד עם איבודים בלתי צפויים של הודעות. מנגנוני ניטור קונבנציונליים המתוארים בספרות כיום יאשימו צמתים תקינים באי שליחת הודעות, ההולכות למעשה לאיבוד בתווך. לבסוף, עבודות קודמות לא כימתו את היחסים אשר צריכים להתקיים בין פרמטרי המערכת, כגון המחיר אותו משלם צומת משתף פעולה, העונש על אי שיתוף פעולה וההחלטה מתי להעניש צומת, על מנת להשיג שיתוף פעולה מלא במחיר (ממוצע) מינימלי לכל צומת.

בתזה זו אנו נתייחס לשלוש הבעיות הנ"ל. אנו נמנף את הדמיון הרב בין פרוטוקולי P2P על-מנת להגדיר ממשק ניטור אבסטרקטי המתאים לזיהוי התנהגות אנוכית במגוון פרוטוקולים מסוג זה. ממשק זה יאפשר לכל צומת לנטר צמתים אחרים ולהחליט מתי להאשים צמתים בהתנהגות אנוכית. לאחר מכן, נציג מנטר אנוכיות אשר עמיד באיבודי הודעות (LTSM) המממש ממשק זה.

המודל שלנו מניח מערכת P2P המורכבת מצמתים כך שכל צומת הוא אנוכי ורציונלי. כלומר, כל צומת רוצה להשתתף בפרוטוקול תוך בחירת אסטרטגיה אשר ממזערת את מחיר השתתפותו. מרחב האסטרטגיות מורכב מההחלטה אילו חבילות להעביר מתוך החבילות הנדרשות על-ידי הפרוטוקול. צומת מוגדר כמשתף פעולה במידה והוא שולח את כל החבילות אותן הוא נדרש לשלוח על-פי הפרוטוקול ולא-משתף פעולה אחרת. לשם פשטות, אנו מניחים כי מחיר שליחת כל החבילות זהה.

צמתים יכולים להצטרף ולהתנתק מהמערכת באופן דינמי. ניתן לנתק צמתים מהמערכת, למשל בשל אנוכיות, וכן ניתן להרשות לצמתים שהואשמו באנוכיות לחזור למערכת אחרי תשלום קנס המוגדר על-ידי האפליקציה. הקנס מוגדר במונחים של מחיר שליחת הודעות. למשל, עבור fine=7 יאלץ צומת חשוד לשלוח שבע חבילות עונשין על-מנת לחזור ולהתחבר למערכת. תשלום עבור התחברות למערכת נדרש במערכות בהן אפשריות התקפות Sybil, כיוון שניתן לנצל לרעה כניסה ללא תשלום במערכות אלו.

אנו מסווגים את ההודעות בפרוטוקול P2P לשתי קטגוריות. הקטגוריה הראשונה מורכבת מהודעות הנשלחות בקצב קבוע וידוע מראש, הודעה אחת לכל יחידת זמן נתונה. דוגמאות לתעבורה מסוג זה הן חבילות חיות (keep-alive) וכן חבילות מידע בשטף קבוע (CBR). שטף קבוע אפשרי בסביבה עם איבודי הודעות אם הפרוטוקול מחייב כל צומת לפצות על חבילות שאבדו, על-ידי שליחת חבילות דמה ריקות במקומן. הקטגוריה השנייה מורכבת מהודעות מונחות בקשות. הודעות אלו נשלחות בתגובה לקבלת הודעות אחרות המגיעות בזמנים בלתי צפויים מראש, כגון העברה של הודעות מידע בפרוטוקול ניתוב או שליחת חתיכת מידע בתגובה לבקשה בפרוטוקול הפצת תוכן.

אנו מניחים כי הרשת מעליה פועל פרוטוקול ה-P2P סובלת מאיבודי הודעות בלתי תלויים ומפולגים בצורה אחידה (i.i.d) בהסתברות איבוד p. חשוב לשים לב כי צומת המזהה איבוד של חבילה אותה הוא בעצמו שלח יכול לחזור על שליחתה על-מנת שלא להיחשד באנוכיות. מסיבה זו נגביל את תשומת לבנו להנחה כי שצומת אינו מסוגל לזהות איבודים של חבילות שהוא בעצמו שולח.

הממשק של מנגנון הניתור שלנו הוא כדלהלן: המתודה start_monitor נקראת על-ידי האפליקציה כאשר צומת חדש נצפה לראשונה או כאשר צומת חשוד מוחזר למערכת לאחר תשלום קנס; המתודה missed_message נקראת כאשר הודעה שהאפליקציה אמורה לקבל אינה מגיעה בזמן והמתודה detected_message נקראת כאשר הודעה זו מתקבלת בזמן. לבסוף, is_selfish מציינת האם צומת מסוים חשוד באנוכיות.

מנגנון ה-LTSM, אשר רץ בכל צומת, שומר רשומה עבור כל צומת מנוטר. צומת נחשד באנוכיות אם הוא שולח פחות מ-τ הודעות בחלון של W הודעות, כאשר τ הוא סף הגילוי. החלונות בפרוטוקול אינם חופפים – כל המונים מתאפסים בסוף כל חלון בגודל W. היחידות של W ו- τ הן מספר הודעות.

אלגוריתם ה-LTSM מתואר באלגוריתם 4.1. הרשומה של כל צומת מנוטר מכילה שני מונים ומשתנה בוליאני, is_selfish, המציין האם הצומת המנוטר נחשד באנוכיות. המונה הראשון, X, סופר את מספר החבילות הצפויות בחלון הנוכחי. המונה השני, Y, סופר את מספר החבילות שהתקבלו בחלון הנוכחי. מתודת ה-start_monitor מאפסת את המונים הנ"ל וקובעת את is_selfish ל-false. מתודת ה-is_selfish מחזירה את המשתנה הבוליאני המתאים. מתודת ה-missed_message מקדמת את החלון על ידי הגדלת המונה X. ההחלטה האם צומת מנוטר נחשד באנוכיות מתקבל בסוף החלון עבור אותו צומת, כאשר המונה X מגיע ל-W. אם הצומת אינו נחשד, אזי כל המונים מאופסים וחלון חדש מתחיל. לבסוף, המתודה detected_message מגדילה את המונה Y, ולאחר מכן מקדמת את החלון כפי שנעשה ב-missed_message. חשוב לשים לב שהמונים אינם מתקדמים עבור צומת חשוד, וכן כי צומת כזה אינו מקבל שרות, אלא אם כן הוא משלם קנס.

אלגוריתם 4.2 מתאר שתי דוגמאות שימוש כלליות של LTSM, דוגמה אחת עבור CBR ודוגמה נוספת עבור תעבורה מונחית בקשות. אנו מניחים כי start_monitor נקראה עבור כל צומת. עבור CBR, האלגוריתם מוודא שמגיעה הודעה בכל פרק זמן מוגדר מראש; detected_message נקראת אם הודעה כזו אכן מגיעה, missed_message נקראת אחרת. באופן דומה, עבור תעבורה מונחית בקשות, מוגדר מרווח זמן מסוים מרגע קבלת הבקשה, כך ש-missed_message נקראת אם התגובה הצפויה אינה מגיעה בזמן זמן ו-detected_message נקראת אחרת. צומת שנחשד באנוכיות אינו מקבל שרות עד למועד תשלום הקנס.

אחת התרומות המרכזיות של תזה זו היא כימות מתמטי של הקשרים בין גודל הקנס לעיל, המחיר של ביצוע פעולה בסיסית (כגון שליחת הודעה בודדת), ההסתברות לאיבוד הודעה, ההחלטה על המועד שבו יש להעניש צומת (ערך הסף לעיל, τ) ותוחלת המחיר עבור צמתים משתפי-פעולה וצמתים שאינם משתפי-פעולה. אנו מראים כיצד ניתן לקבוע את הפרמטרים של LTSM על-מנת ששיתוף פעולה מלא ודבקות בפרוטוקול יהוו נקודת שיווי משקל Nash, תוך כדי מינימיזציה של תוחלת המחיר של צמתים תקינים אשר משתפים פעולה.

לבסוף אנו מראים שימושים של ממשק הניטור המתואר לעיל בפרוטוקולים קיימים להפצת מידע (multicast) ובפרוטוקולים לניתוב ברשתות על-חוטיות ניידות (MANET). שימוש ב-LTSM עם הפרמטרים המתאימים מאפשר להוסיף באופן אוטומטי תמיכה בהתמודדות עם איבודי הודעות למנגנוני ההתמודדות עם התנהגות אנוכית בפרוטוקולים הנ"ל.