Stochastic Analysis of A Randomized Detection Algorithm for Pollution Attack in P2P Live Streaming Systems

Yongkun Li John C.S. Lui
The Chinese University of Hong Kong

Abstract

Pollution attack is known to have a disastrous effect on existing P2P infrastructures: it can reduce the number of legitimate P2P users by as much as 85%, and it generates abundant bogus data which may deplete the communication bandwidth. We propose a distributed defense and detection mechanism to resolve pollution attacks. The mechanism is composed of a set of "randomized" and "fully distributed" algorithms that can be executed by any legitimate peer. We present the analytical framework to quantify (a) the probability of false negative, (b) the probability of false positive, and (c) the distribution of time needed for detection. In our detection algorithm and analysis, we consider the case of (1) single attacker within the neighborhood, (2) multiple attackers within the neighborhood. Furthermore, we show how to "optimize" the system parameters so as to quickly discover and eliminate malicious peers from the system.

1. Introduction

In the past few years, we have witnessed the use of data-driven mesh-pull technique to provide large scale P2P video streaming services. There are number of such P2P video streaming services[1, 2, 3], which offer live streaming as well as video-on-demand services[4]. These services use the P2P technology to distribute video content to hundreds of thousands of concurrent users to achieve a highly scalable content distribution system.

However, the distributed nature of these P2P streaming services makes them vulnerable to many security attacks. One devastating security attack is via *stream pollution*[5]. Under this attack, a misbehaving peer uploads bogus data to its neighboring peers. This not only degrades and damages the quality of playback, but more importantly, these bogus data will be forwarded by peers and pollute the entire P2P streaming system. There are many causes for the pollution attack. One of them is that a P2P software is mis-configured and it uploads polluted information to other peers. Another possibility is that legitimate content owners want to stop the illegal distribution of their movies by some P2P companies, and these content owners may hire a pollution agent to

carry out such attack. As reported in [5], pollution attack can reduce the number of legitimate peers by as much as 85%. This creates a devastating effect to the streaming system, and at the same time, the bandwidth used in exchanging the polluted data will be wasted, and lowering the efficiency of the networks.

In this paper, we present a distributed method to *defend* and *detect* malicious peers in pollution attacks of P2P live streaming systems. Any legitimate peer can execute the detection algorithm to discover malicious peers within its neighborhood. We present the analytical framework for such distributed detection procedure. In particular, we consider performance measures such as (a) probability of false positive, (b) probability of false negative and, (c) the distribution on time needed to identify all attackers in a P2P streaming system. We also illustrate how one can *optimize* the system so as to quickly discover malicious peers. We like to point out that the evaluation methodology is quite general and it can be applied to quantify misbehavior in other interactive communication systems such as wireless mesh networks and online social networks.

The outline of the paper is as follows. In Section 2, we briefly present pollution attacks in P2P live streaming systems as well as the defense mechanism. In Section 3, we present the methodology to detect the pollution attackers. We first consider the case that there is a single attacker within the neighborhood of a legitimate peer, the performance evaluation of this case is presented in Section 4. In Section 5, we consider the scenario of multiple attackers, the associated detection mechanism and its performance. In Section 6, we validate our analysis via extensive simulation. Based on the mathematical models, we present in Section 7 how one can *optimize* the detection so as to quickly discover malicious peers. Section 8 provides the related work and Section 9 concludes.

2. Pollution Attacks and Defense Method

A P2P video streaming is similar to the BitTorrent (BT) file sharing system: a peer needs to upload data that it has received from its neighbors. The server of a P2P streaming system which generates a video stream needs to first divides the video segment into non-overlapping *chunks* (e.g., 2M bytes in PPLive). A chunk is further divided into non-overlapping *pieces* (e.g., 16 KBytes), while a piece is further divided into non-overlapping *sub-pieces* (e.g., 1 KBytes). Each peer maintains a bitmap, when a peer has a particular chunk, its bitmap of that chunk will be "1" to indicate its availability. Peers periodically exchange their bitmaps so that they know which chunks its neighboring peers are holding. Piece, on the other hand, is a unit of media playback and it is a function of the video encoding method. Lastly, sub-piece is a unit of data transfer between peers. In summary, the separation of chunks, pieces and sub-pieces is to balance between communication/memory overheads and playback requirement.

To obtain a particular piece, a peer needs to seek download service from its neighboring peers. Only when the downloaded piece is correct, a peer can then perform the playback operation. Another point worth mentioning is that unlike the BT systems wherein neighboring peers are selected via the "*tit-for-tat*" policy, neighboring peers in P2P live streaming systems are relatively *stable*

because of the following reasons: (1) the system (i.e., PPLive, PPStream, etc) is usually proprietary and users cannot modify the software behavior; (2) peers have the same objective to watch the movie with low delay/jitter so they are willing to contribute their upload bandwidth resource.

Pollution attack of P2P streaming systems was first reported in [6]. An attacker can upload bogus sub-pieces to its neighboring peers. To reduce the chance of being detected, an attacker can even choose to "*imitate*" a good peer: for some uploads, the attacker provides valid sub-pieces to its neighbors and for other uploads, it provides bogus sub-pieces. When a legitimate peer receives these bogus sub-pieces, it may not be able to perform a smooth playback of the related piece, and worse yet, the bogus sub-pieces may crash the streaming software. Note that due to the P2P communication paradigm, bogus sub-pieces are also forwarded to other peers, creating an avalanche of P2P software failure.

To counter the the above pollution attack, the following defense mechanism can be used. The defense mechanism consists of two components: (1) error concealment and; (2) verification.

For the error concealment, the server uses the error correction method[7]. Specifically, for each piece the server generates, the server divides the piece into m non-overlapping sub-pieces. Furthermore, the server also generates $n - m$ (for $n \geq m$) independently redundant sub-pieces. For each piece, the server also generates a hash code and this hash value is appended into each associated sub-piece. The server then uploads these n sub-pieces to peers in the system. To obtain a piece, a peer needs to request its neighbors for the corresponding sub-pieces. A peer only needs to obtain any m sub-pieces from its neighbors so as to decode the original piece. The validity of a piece can be verified by computing its hash value from these m sub-pieces and checking with the appended hash code. When the hash value matches, the peer can playback that piece. If it does not match, the peer can request for additional sub-piece from its neighbors, or simply drop that piece for playback.

To avoid an attacker from pretending as the server in performing the above operation, we have the verification procedure that the server signs all hash codes it generates. In particular, for every l pieces that the server generates, the server signs with its signature and distributes this information to peers via gossiping protocols. Via this way, peers can be certain that the hash code for each piece is indeed valid and these hash codes are generated by a trusted server.

When a peer discovers the computed hash value of a given piece does not match with the appended hash code, it knows that there are potential attackers in its neighboring peers. It is not only beneficial for this peer to *discover* these attackers and blacklist them from further data exchange, but the peer can also inform the system the existence of these attacking peers to further limit their damage. In this paper, we propose a performance evaluation methodology to address: (1) *the accuracy of the detection method*, and (2) *the duration needed to detect these attackers*.

3. Models for Detection Methods

In this section, we first provide a formal description of the detection methodology. In this paper, we consider several attack scenarios and for each scenario, we derive the performance measures. The P2P system is modeled as an undirected graph $G = (V, E)$, where each node in V corresponds to a peer in a P2P system. If there is an edge $(i, j) \in E$ which connects peer i and j, it implies that peer i and j are neighbors and they can upload/download subpieces from each other. The set of neighbors of peer i is denoted by $\mathcal{N}^i$, where $\mathcal{N}^i = \{j \in V | (i, j) \in E\}$. Without loss of generality, we have $|\mathcal{N}^i| = N$.

We model the interaction among peers as a *repeated process*, and time proceeds in rounds $t=1, 2, 3, ...$ Let us focus on a particular legitimate peer, say i. Peers in $\mathcal{N}^i$ may choose from two possible actions, (a) upload a valid sub-piece to peer i, or (b) upload a bogus sub-piece to peer i. If it is a legitimate peer, it will choose the first action, while a malicious peer can choose either of these actions to damage the system and to reduce the possibility to be detected. Denote $\mathcal{T}^i(t)$ as the subset of good neighbors for peer i at round t, in other words, all peers in $\mathcal{T}^i(t)$ are trustable. Correspondingly, $\bar{\mathcal{T}}^i(t)$ is the set of potentially untrustable neighbors. Note that it is possible that some good neighboring peers may belong to $\bar{\mathcal{T}}^i(t)$. Obviously, we have

$$\mathcal{N}^i = \mathcal{T}^i(t) \cup \bar{\mathcal{T}}^i(t) \quad \forall t.$$

Since we consider a repeated process, and in each round t, we have a untrustable set $\bar{\mathcal{T}}^i(t)$ which contains all potential malicious neighbors of peer i. After a finite number of rounds, the *intersection* of such untrustable sets will only contain the malicious peers with high probability. We can then quantify, with high accuracy, the false negative and false positive of a neighbor being a good or malicious peer. We define $\mathcal{M}^i(t)$ as the intersection of the untrustable sets of peer i until time t, or formally as: $\mathcal{M}^i(t) = \cap_{r=1}^{t} \bar{\mathcal{T}}^i(r)$ for $t = 1, 2, \ldots$. When we initialize $\mathcal{M}^i(0) = \mathcal{N}^i$, we then have:

$$\mathcal{M}^i(t) = \mathcal{M}^i(t-1) \cap \bar{\mathcal{T}}^i(t) \qquad t = 1, 2, \ldots \tag{1}$$

Eq. (1) shows the general detection algorithm for peer i at round t. As long as t is sufficiently large, the set $\mathcal{M}^i(t)$ will only contain malicious neighbors of peer i, then we can say, with high accuracy, that all malicious neighbors of peer i are detected. This general detection process is illustrated in Fig. 1. Note that as we proceed in rounds, the size of $\mathcal{M}^i(t)$ will shrink, or $|\mathcal{M}^i(t)| \leq |\mathcal{M}^i(t-1)|$. Since this detection process is used by all legitimate peers, $\cup_{i \in V} \mathcal{M}^i(t)$ contains all malicious peers in a P2P network. Now, the technical challenge is how to divide the set of neighbors into two subsets at each round and guarantee that all malicious peers remain in $\bar{\mathcal{T}}^i(t)$. In the following sections, we give detailed algorithms on how to perform malicious peer detection.

4. Single Malicious Peer in $\mathcal{N}^i$

Let us focus on a particular legitimate peer, say i. We assume that it has N neighbors, or $|\mathcal{N}^i| = N$. Among these neighbors, there is only a single malicious

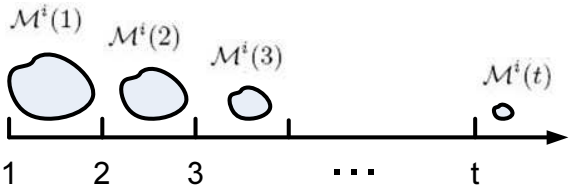


Figure 1: General detection method where the set $\mathcal{M}^i(t)$ gets smaller: $|\mathcal{M}^i(t)| \leq |\mathcal{M}^i(t-1)|$ for round $t = 1, 2, ...$

peer and this peer will choose to upload a polluted sub-piece with probability $1 - \delta$. When $0 < \delta < 1$, the attacker may choose to imitate as a legitimate peer in some of the uploads. When peer i needs a particular piece, it downloads the corresponding sub-pieces from its neighbors in $\mathcal{N}^i$. Since the number of neighbors, $|\mathcal{N}^i| = N$, is usually much larger than m, the minimum decoding requirement of a piece. So peer i needs to decide which neighbors to download. For P2P streaming, load balancing is important and we model it as a random peer selection strategy: each neighbor in $\mathcal{N}^i$ will be selected with the same probability α. If a neighboring peer in $\mathcal{N}^i$ is selected, it uploads the sub-piece to peer i. We use the following notation to describe the detection algorithm:

$\mathcal{M}^i(t)$: the set of potentially malicious neighbors of peer i at the end of round t. We have $\mathcal{M}^i(0) = \mathcal{N}^i$.
$U^i(t)$: the set of neighbors of peer i which upload to peer i at round t.
$\bar{U}^i(t)$: the set of neighbors of peer i which do not upload to peer i at round t. We have $U^i(t) \cup \bar{U}^i(t) = \mathcal{N}^i$.
α : probability that a peer in $\mathcal{N}^i$ is chosen to upload.
δ : probability that a malicious peer uploads valid sub-pieces, or the probability of imitation.

To present the detection algorithm, we consider two cases: (1) the imitation probability $\delta = 0$; (2) the imitation probability $\delta > 0$.

Single attacker with imitation probability $\delta = 0$: In this case, the malicious peer always upload polluted sub-pieces. The detection algorithm is as follows: before detection, peer i initializes $\mathcal{M}^i(0) = \mathcal{N}^i$. As peer i proceeds in different rounds, peers in $\mathcal{N}^i$ are partitioned into different types according to their actions and behaviors. When the set of potential malicious peers $\mathcal{M}^i(t)$ shrinks to a singleton, then peer i can claim that it finds the malicious peer. The detection process at round t is:

Algorithm A1: Detection Algorithm for Single Attacker and $\delta = 0$

If (the computed hash value is the same as the appended hash code):
$\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1) \cap \bar{U}^i(t)$;
else: $\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1) \cap U^i(t)$;

When the hash value matches, it implies that the attacker cannot be in $U^i(t)$. On the other hand, if mismatch is found, then the attacker must be in $U^i(t)$. Note that the set $\mathcal{M}^i(t)$ shrinks as we proceed with more rounds, and eventually, $\mathcal{M}^i(t)$ only contains the attacker.

Let us quantify the performance of this detection process. Note that the initial condition is $\mathcal{M}^i(0) = \mathcal{N}^i$ and peer i executes the detection algorithm with sufficient number of rounds to shrink the set $\mathcal{M}^i(t)$ until it reaches to the state that all peers in $\mathcal{M}^i(t)$ are malicious with high probability. Note that when the number of rounds is not sufficiently large, it is possible that some peers in $\mathcal{M}^i(t)$ are not malicious peers but legitimate peers. We define a variable to indicate the probability that a randomly chosen peer in $\mathcal{M}^i(t)$ is not malicious. This is in fact *the probability of false positive*, and we denote it as $\mathcal{P}^i_{fp}(t)$. In some scenarios, malicious peers may be wrongly removed from $\mathcal{M}^i(t)$ (as we will see it in later scenarios). Therefore, we define *the probability of false negative* as $\mathcal{P}^i_{fn}(t)$. The third performance measure is to quantify the number of rounds peer i needs to detect the malicious peer. Let $\mathcal{R}$ be the random variable denoting the number of rounds we need to shrink $\mathcal{M}^i(t)$ until it reaches the state of only containing malicious peers and we are interested in $E[\mathcal{R}]$. For the performance measures of $\mathcal{P}_{fn}(t)$ and $\mathcal{P}_{fp}(t)$, we derive them later, and we only focus on the performance measure of $\mathcal{R}$ here.

Lemma 1. *For the scenario of single attacker with imitation probability $\delta = 0$, if Algorithm A1 is employed, then the cumulative distribution of $\mathcal{R}$ is:* $P(\mathcal{R} \leq r) = \sum_{j=0}^{r} \binom{r}{j} \alpha^j (1-\alpha)^{r-j} \left[1 - \alpha^j (1-\alpha)^{r-j}\right]^{N-1}$.

Proof: Note that, in this scenario, there is only one attacker and it stays in the malicious peer set. If a legitimate peer wants to be isolated from the malicious peer set, it must be chosen differently from the attacker in some round, or

$$\begin{aligned} P(\mathcal{R} \leq r) &= P(\text{after r rounds, } |\mathcal{M}^i(r)| = 1) \\ &= P(\text{for each good peer in } \mathcal{N}^i, \text{ it's selected differently} \\ &\quad \text{from the malicious peer in at least one of the } r \text{ rounds}) \\ &= \sum_{j=0}^{r} \binom{r}{j} \alpha^j (1-\alpha)^{r-j} \left[1 - \alpha^j (1-\alpha)^{r-j}\right]^{N-1}. \blacksquare \end{aligned}$$

Based on Lemma 1, the expected number of rounds to detect the malicious peer can be easily derived as follows:

$$E[\mathcal{R}] = \sum_{r=1}^{\infty} r P(\mathcal{R}=r) = \sum_{r=1}^{\infty} r \left[P(\mathcal{R} \leq r) - P(\mathcal{R} \leq r-1)\right]. \tag{2}$$

Single attacker with imitation probability $\delta > 0$: In this case, the malicious peer can choose to upload a valid sub-piece with probability $\delta > 0$ so as to lower the chance of being detected. For peer i, when it receives sufficient

number of sub-pieces and recovers the original piece with a matched hash value, peer i cannot shrink the set $\mathcal{M}^i(t)$. In other words, set shrinkage only occurs when the malicious peer uploads a polluted sub-piece. Peer i will execute the following "*baseline*" detection algorithm.

Algorithm A2: Baseline Detection Alg. for Single Attacker and $\delta > 0$

If (the computed hash value is the same as the appended hash code):
$\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1)$;
else: $\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1) \cap U^i(t)$;

Note that for this algorithm, we can only shrink $\mathcal{M}^i(t)$ when the computed hash value does not match with the appended hash code. Let us quantify the performance of Algorithm A1 and A2 in the following lemma.

Lemma 2. *For Algorithm A1 and A2, the probability of false negative* $\mathcal{P}_{fn}(t) = 0$ *and the probability of false positive* $\mathcal{P}_{fp}(t) = \frac{|\mathcal{M}^i(t)|-1}{|\mathcal{M}^i(t)|}$. *For Algorithm A2,* $\mathcal{R}$ *follows the probability mass function of*
$P(\mathcal{R}=r)=\sum_{u=1}^{r}\binom{r-1}{u-1}[\alpha(1-\delta)]^u[1-\alpha(1-\delta)]^{r-u}\left[(1-\alpha^u)^{N-1}-(1-\alpha^{u-1})^{N-1}\right]$.

Proof: Since $\mathcal{M}^i(0) = \mathcal{N}^i$, and according to these two detection algorithms, the malicious peer always remains in $\mathcal{M}^i(t)$, so the probability of false negative is just 0, i.e., $\mathcal{P}_{fn}(t) = 0$. To compute the probability of false positive, note that, there is only one malicious peer in $\mathcal{N}^i$, and it remains in the malicious peer set $\mathcal{M}^i(t)$, so the probability of false positive can be computed as $\mathcal{P}_{fp}(t) = \frac{|\mathcal{M}^i(t)|-1}{|\mathcal{M}^i(t)|}$.

Let us consider $\mathcal{R}$, the number of rounds needed to shrink the set $\mathcal{M}^i(t)$ to a singleton. Note that for some rounds, $\mathcal{M}^i(t)$ cannot be shrunk and this occurs when the computed hash value matches with the appended hash code. The only "*useful*" rounds are those that hash value does not match . Let $\mathcal{U}$ be the random variable denoting the number of these useful rounds. Obviously, $\mathcal{U}$ is stochastically smaller than $\mathcal{R}$, or $\mathcal{U} \preceq_{st} \mathcal{R}$. To compute the distribution of $\mathcal{U}$, observe that each of the legitimate peers remains in the potential malicious set $\mathcal{M}^i(t)$ as long as it uploads in each useful round, and the legitimate peer will upload in each round independently with probability α. So each legitimate peer remains in the set for a geometrically distributed number of useful rounds, or

$$\begin{aligned} P(\mathcal{U} \le u) &= P(\text{after } u \text{ useful rounds, the potential} \\ &\quad \text{malicious peer set shrinks to a singleton set}) \\ &= P(\text{each legitimate peer in } \mathcal{N}^i \\ &\quad \text{does not upload in at least one useful round}) \\ &= (1-\alpha^u)^{N-1}. \end{aligned}$$

To compute the distribution of $\mathcal{R}$, observe that a useful round happens when the malicious peer uploads polluted sub-pieces, and the probability of this event

is $\alpha(1-\delta)$. Given the number of useful rounds $\mathcal{U}$, the conditional distribution $P(\mathcal{R}=r|\mathcal{U}=u)$ is a negative binomial distribution, so we have:

$$P(\mathcal{R}=r)=\sum_{u=1}^{r} P(\mathcal{U}=u)P(\mathcal{R}=r|\mathcal{U}=u)$$
$$=\sum_{u=1}^{r}\binom{r-1}{u-1}[\alpha(1-\delta)]^u[1-\alpha(1-\delta)]^{r-u}P(\mathcal{U}=u). \quad (3)$$

By substituting $P(\mathcal{U}=u)$, we obtain the result stated in Lemma 2. ∎
Based on Lemma 2, $E[\mathcal{R}]$ can be easily derived as $E[\mathcal{R}]=\sum_{r=1}^{\infty} rP(\mathcal{R}=r)$.

Randomized Detection Algorithm: Note that the detection algorithm A2 cannot shrink the size of $\mathcal{M}^i(t)$ when the computed hash value matches with the appended hash code, and this causes $\mathcal{R}$ to be unnecessarily large. It is important to note that there are two causes to have a matched hash value: (1) the neighboring peers which are selected to upload are all legitimate peers; (2) the malicious peer is selected to upload and it decides to upload a valid sub-piece, and this occurs with probability $\alpha\delta$. This probability, in general, is quite small since the ultimate objective of the attacker is to maximize the damage to the system. In here, we propose a *randomized detection algorithm* by considering those rounds in which the hash code matches. Peer i will consider these rounds as useful and shrink $\mathcal{M}^i(t)$ with probability $1-p$. The detection algorithm is:

Algorithm A3: Randomized Detection for Single Attacker and $\delta > 0$

If (the computed hash value is the same as the appended hash code):
with probability p : $\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1)$;
with probability $1-p$: $\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1) \cap \bar{U}^i(t)$;
else: $\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1) \cap U^i(t)$;

When the malicious peer pretends to be a good peer in a round and if we take that round as useful for detection, then the malicious peer will be removed from $\mathcal{M}^i(t)$. In other words, $\mathcal{P}_{fn}$ is *not* zero any more. Moreover, even if $\mathcal{M}^i(t)$ shrinks to a singleton, we cannot conclude that it is the malicious peer. Therefore, $\mathcal{P}_{fp}$ cannot be computed as before. In the following, we derive these performance measures and summarize them in lemma 3.

Lemma 3. *When Algorithm A3 is used after t rounds, we have $\mathcal{P}_{fn}(t) = 1-(1-\alpha\delta(1-p))^t$, $\mathcal{P}_{fp}(t) = \frac{(N-1)(1-\alpha(1-\delta)(1-\alpha)-\alpha(1-\alpha+\alpha\delta)(1-p))^t}{(1-\alpha\delta(1-p))^t+(N-1)(1-\alpha(1-\delta)(1-\alpha)-\alpha(1-\alpha+\alpha\delta)(1-p))^t}$, and the number of detection rounds $\mathcal{R}$ follows the distribution of $P(\mathcal{R}=r) = \sum_{u=1}^{r}\binom{r-1}{u-1}(p_u)^u(1-p_u)^{r-u}\left[\sum_{j=0}^{u}\binom{u}{j}q^j(1-q)^{u-j}\left[1-\alpha^j(1-\alpha)^{u-j}\right]^{N-1}-\sum_{j=0}^{u-1}\binom{u-1}{j}q^j(1-q)^{u-1-j}\left[1-\alpha^j(1-\alpha)^{u-1-j}\right]^{N-1}\right]$, where $p_u = \alpha(1-\delta)+(1-\alpha+\alpha\delta)(1-p)$ and $q=\frac{\alpha(1-\delta)}{\alpha(1-\delta)+(1-\alpha+\alpha\delta)(1-p)}$.*

Proof: Note that when Algorithm A3 is used, the malicious peer may be removed from $\mathcal{M}^i(t)$ in round t with probability $\alpha\delta(1-p)$. So after t rounds, the

probability that the malicious peer is not in the set $\mathcal{M}^i(t)$, which is also called the probability of false negative, is $\mathcal{P}_{fn}(t) = 1 - (1 - \alpha\delta(1-p))^t$.

To derive $\mathcal{P}_{fp}(t)$, note that, the malicious peer will be removed from $\mathcal{M}^i(t)$ at round t if and only if it uploads valid sub-pieces and that round is not being ignored, which happens with probability $P_M = \alpha\delta(1-p)$. On the other hand, a legitimate peer may also be removed from $\mathcal{M}^i(t)$, and the probability is $P_G = \alpha(1-\delta)(1-\alpha) + (1-\alpha+\alpha\delta)\alpha(1-p)$. $\mathcal{P}_{fp}(t)$ can be derived as follows:

$$\begin{aligned}
\mathcal{P}_{fp}(t) &= P(j \text{ is a randomly chosen legitimate peer} \mid j \in \mathcal{M}^i(t)) \\
&= 1 - P(j \text{ is a malicious peer} \mid j \in \mathcal{M}^i(t)) \\
&\approx 1 - P(j \text{ is a malicious peer} \mid j \text{ is in } \mathcal{M}^i(t) \text{ for } t \text{ rounds}) \\
&= 1 - \frac{P(j \text{ is a malicious peer and in } \mathcal{M}^i(t) \text{ for } t \text{ rounds})}{P(j \text{ is in } \mathcal{M}^i(t) \text{ for } t \text{ rounds})}. \qquad (4)
\end{aligned}$$

Applying the theorem of total probability, we have

$$\begin{aligned}
\mathcal{P}_{fp}(t) &= 1 - \frac{\frac{1}{N}(1-P_M)^t}{\frac{1}{N}(1-P_M)^t + (1-\frac{1}{N})(1-P_G)^t} \\
&= \frac{(N-1)(1-\alpha(1-\delta)(1-\alpha)-\alpha(1-\alpha+\alpha\delta)(1-p))^t}{(1-\alpha\delta(1-p))^t + (N-1)(1-\alpha(1-\delta)(1-\alpha)-\alpha(1-\alpha+\alpha\delta)(1-p))^t}.
\end{aligned}$$

To derive the probability distribution of $\mathcal{R}$, we first derive the distribution of $\mathcal{U}$. Observe that, in each useful round, there are two possible cases. The first case is that peer i perceives that the malicious peer uploaded in that round, which happens when the malicious peer uploaded polluted sub-pieces, and the probability is $\alpha(1-\delta)$. The second case is that peer i perceives that the malicious peer did not upload, which happens when the malicious peer did not upload or only uploaded valid sub-pieces, and the probability is $\alpha\delta(1-p) + (1-\alpha)(1-p)$. So, in a useful round, the first case happens with probability $q = \frac{\alpha(1-\delta)}{\alpha(1-\delta)+\alpha\delta(1-p)+(1-\alpha)(1-p)}$, In each round, a legitimate peer will be removed from $\mathcal{M}^i(t)$ if and only if it is not chosen to upload when the first case occurs, or it is chosen to upload when the second case occurs. Therefore:

$$\begin{aligned}
P(\mathcal{U} \le u) &= P(\text{after } u \text{ useful rounds, peers in } \mathcal{M}^i(u) \text{ are all potentially malicious}) \\
&= P(\text{for each good peer in } \mathcal{N}^i, \text{ it must act differently} \\
&\quad \text{from the malicious peer for at least one useful round}) \\
&= \sum_{j=0}^{u} \binom{u}{j} q^j (1-q)^{u-j} \left[1 - \alpha^j (1-\alpha)^{u-j}\right]^{N-1}.
\end{aligned}$$

To derive the distribution of $\mathcal{R}$, note that the useful round will happen with probability $p_u = \alpha(1-\delta)+\alpha\delta(1-p)+(1-\alpha)(1-p)$. The conditional distribution

$P(\mathcal{R}=r|\mathcal{U}=u)$ is a negative binomial distribution, or

$$P(\mathcal{R}=r) = \sum_{u=1}^{r} P(\mathcal{U}=u)P(\mathcal{R}=r|\mathcal{U}=u)$$
$$= \sum_{u=1}^{r} \binom{r-1}{u-1} (p_u)^u (1-p_u)^{r-u} \left[P(\mathcal{U}\leq u) - P(\mathcal{U}\leq u-1)\right]. \quad (5)$$

By substituting the distribution of $\mathcal{U}$, we get the distribution of $\mathcal{R}$ as stated in the lemma. and $E[\mathcal{R}]$ can be easily computed. ∎

5. Multiple Malicious Peers in $\mathcal{N}^i$

In this section, we consider the general scenario that there are $k > 1$ malicious attackers in the neighborhood set $\mathcal{N}^i$.

Multiple attackers with imitation probability $\delta = 0$: We first consider the case that all malicious peers in $\mathcal{N}^i$ always upload polluted sub-pieces to peer i whenever they are selected for data exchange. In the single attacker case, we partition the neighboring peers into two sets at each round. One is the set for peers who upload, the other is the set for peers who do not upload. Moreover, by verifying the hash code, peer i knows which set a malicious peer is in such that $\mathcal{M}^i(t)$ can be shrunk and eventually it will only contain the malicious peer. However, when multiple malicious peers are in $\mathcal{N}^i$, such identification is more challenging. Since in any round, if some of the malicious peers upload but other malicious peers do not, then by knowing that the computed hash value does not match with the appended hash code, peer i cannot claim that *all* malicious peers are in the uploading set. On the other hand, since all malicious peers upload polluted sub-piece (because $\delta = 0$), therefore, when the hash value matches, we can be certain that all malicious peers are in the non-uploading set $\bar{U}^i(t)$ and we can shrink $\mathcal{M}^i(t)$. We initialize $\mathcal{M}^i(0) = \mathcal{N}^i$ and the detection is as follows.

Alg. B1: Detection Algorithm for Multiple Malicious Peers and $\delta = 0$

If (the computed hash value is the same as the appended hash code):
$\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1) \cap \bar{U}^i(t)$;
else: $\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1)$;

Lemma 4. *If Algorithm B1 is used, then after t rounds, we have* $\mathcal{P}_{fn}(t) = 0$, $\mathcal{P}_{fp}(t) = \frac{|\mathcal{M}^i(t)|-k}{|\mathcal{M}^i(t)|}$, *and* $\mathcal{R}$ *has the distribution of* $P(\mathcal{R}=r) = \sum_{u=1}^{r} \binom{r-1}{u-1}(1-\alpha)^{ku}(1-(1-\alpha)^k)^{r-u}\left[(1-(1-\alpha)^u)^{N-k} - (1-(1-\alpha)^{u-1})^{N-k}\right]$.

Proof: In this algorithm, the possible malicious peers set $\mathcal{M}^i(t)$ shrinks only when all malicious peers do not upload, and peers which are isolated in those rounds are the legitimate peers who upload valid sub-pieces. Thus all malicious

peers remain in $\mathcal{M}^i(t)$ for all rounds, so $\mathcal{P}_{fn}(t) = 0$. On the other hand, since there are k malicious peers, and they all remain in the malicious peers set $\mathcal{M}^i(t)$, so the probability of false positive is simply $\frac{|\mathcal{M}^i(t)|-k}{|\mathcal{M}^i(t)|}$.

To derive the distribution of $\mathcal{R}$, we fist derive the distribution of $\mathcal{U}$. Observe that, all malicious peers do not upload in every useful round (i.e., rounds that hash value matches). So for any legitimate peer, it will be isolated from $\mathcal{M}^i(t)$ as long as it uploads in at least one useful round. We have

$$\begin{aligned} P(\mathcal{U} \leq u) &= P(\text{after } u \text{ useful rounds, peers in the} \\ &\quad \text{malicious set } \mathcal{M}^i(u) \text{ are all malicious peers}) \\ &= P(\text{for each legitimate peer in } \mathcal{N}^i, \text{ it must upload} \\ &\quad \text{in at least one useful round}) = (1-(1-\alpha)^u)^{N-k}. \end{aligned}$$

To derive the distribution of $\mathcal{R}$, observe that, the useful round will happen with probability $(1-\alpha)^k$. Given the number of useful rounds $\mathcal{U}$, the conditional distribution $P(\mathcal{R} = r|\mathcal{U} = u)$ is a negative binomial distribution. We then have

$$\begin{aligned} P(\mathcal{R} = r) &= \sum_{u=1}^{r} P(\mathcal{U} = u) P(\mathcal{R} = r|\mathcal{U} = u) \\ &= \sum_{u=1}^{r} \binom{r-1}{u-1} (1-\alpha)^{ku} (1-(1-\alpha)^k)^{r-u} P(\mathcal{U} = u). \end{aligned} \tag{6}$$

By substituting the distribution of $P(\mathcal{U} = u)$, we obtain the claimed results. ■

Based on lemma 4, the average number of rounds can be easily computed as $E[\mathcal{R}] = \sum_{r=1}^{\infty} rP(\mathcal{R} = r)$.

Multiple attackers with imitation probability $\delta > 0$: Let us now consider the case of multiple malicious peers, each of which may upload a valid sub-piece with an imitation probability $\delta > 0$. Under this situation, if the computed hash value is different from the appended hash code, peer i cannot be certain that *all* malicious peers are in the upload set or not. On the other hand, if the computed hash value matches, peer i is still confronted with the problem of accurately detecting the malicious peers since they may pretend to be good peers. As discussed before, the goal of the malicious peers is to damage the system and to reduce the chance to be detected, so δ cannot be too large. To this end, we propose the following *randomized detection algorithm*:

Alg. B2: Detection Algorithm for Multiple Malicious Peers and $\delta > 0$

If (the computed hash value is the same as the appended hash code):
with probability p : $\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1)$;
with probability $1-p$: $\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1) \cap \bar{U}^i(t)$;
else: $\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1)$;

Lemma 5. *If Algorithm B2 is used, then after t rounds, we have* $\mathcal{P}_{fn}(t) = 1-\left[1-\alpha\delta(1-p)(\alpha\delta+1-\alpha)^{k-1}\right]^t$, $\mathcal{P}_{fp}(t) = 1-\frac{k}{k+(N-k)(\frac{1-\alpha(1-p)(\alpha\delta+1-\alpha)^k}{1-\alpha\delta(1-p)(\alpha\delta+1-\alpha)^{k-1}})^t}$, *and the number of rounds $\mathcal{R}$ follows the distribution of* $P(\mathcal{R}=r)=\sum_{u=1}^{r}\binom{r-1}{u-1}\left[(1-p)(1-\alpha+\alpha\delta)^k\right]^u\left[1-(1-\alpha+\alpha\delta)^k(1-p)\right]^{r-u}\times \left[(1-(1-\alpha)^u)^{N-k}-(1-(1-\alpha)^{u-1})^{N-k}\right]$.

Proof: To derive $\mathcal{P}_{fn}$, observe that, a malicious peer will be removed from $\mathcal{M}^i(t)$ only when it uploads valid sub-pieces and all other malicious peers do not upload polluted sub-pieces, moreover, that round is not ignored, so this probability is $P_M = \alpha\delta(1-p)(\alpha\delta+1-\alpha)^{k-1}$, where k is the number of malicious peers in $\mathcal{N}^i$. Based on this probability, $\mathcal{P}_{fn}$ can be easily derived as $\mathcal{P}_{fn}(t) = 1-\left[1-\alpha\delta(1-p)(\alpha\delta+1-\alpha)^{k-1}\right]^t$. Note that when peer i executes Algorithm B2 for sufficient rounds, $\mathcal{P}_{fn}\to 1$ as $t\to\infty$. This is obviously not desirable. We will show how to improve this performance measure later.

Now, note that, a legitimate peer will be removed from $\mathcal{M}^i(t)$ only when it uploads in a round that the hashed value matches and at the same time, that round is chosen for detection. This happens with probability $P_G = \alpha(1-p)(\alpha\delta+1-\alpha)^k$. Obviously, $P_M < P_G$. By employing the method in Lemma 3, we can derive the probability of false positive $\mathcal{P}_{fp}(t)$ as follows:

$$\mathcal{P}_{fp}(t) = 1-\frac{\frac{k}{N}(1-P_M)^t}{\frac{k}{N}(1-P_M)^t+(1-\frac{k}{N})(1-P_G)^t} = 1-\frac{k}{k+(N-k)(\frac{1-P_G}{1-P_M})^t}. \quad (7)$$

By substituting P_G and P_M, we get the result stated in the lemma.

Let us derive the distribution of $\mathcal{R}$, and we first focus on the distribution $\mathcal{U}$. Observe that for any malicious peer which is still in $\mathcal{M}^i(t)$, it does not upload in every useful round. As for any legitimate peer, it will be removed from $\mathcal{M}^i(t)$ as long as it uploads in at least one useful round. Therefore

$$\begin{aligned} P(\mathcal{U}\le u) &= P(\text{after } u \text{ useful rounds, peers in the} \\ &\qquad \text{malicious peer set } \mathcal{M}^i(u) \text{ are all malicious }) \\ &= P(\text{for each legitimate peer in } \mathcal{N}^i, \text{ it must upload in} \\ &\qquad \text{at least one useful round}) = \left[1-(1-\alpha)^u\right]^{N-k}. \end{aligned}$$

To derive the distribution of $\mathcal{R}$, note that the useful round happens with probability $(1-\alpha+\alpha\delta)^k(1-p)$. Given the number of useful rounds $\mathcal{U}$, the conditional distribution $P(\mathcal{R}=r|\mathcal{U}=u)$ is a negative binomial distribution.

$$\begin{aligned} P(\mathcal{R}=r) &= \sum_{u=1}^{r} P(\mathcal{U}=u)P(\mathcal{R}=r|\mathcal{U}=u) \\ &= \sum_{u=1}^{r}\binom{r-1}{u-1}\left[(1-p)(1-\alpha+\alpha\delta)^k\right]^u\left[1-(1-\alpha+\alpha\delta)^k(1-p)\right]^{r-u}P(\mathcal{U}=u). \end{aligned}$$

Substituting the distribution $P(\mathcal{U}=u)$, we get the result stated in the lemma. ∎

In the previous discussion, we present the detection algorithms at one round for different scenarios, i.e., we focus on how to shrink $\mathcal{M}^i(t)$ for round t. Let us now discuss the termination condition: how many rounds we need to execute so as to have high guarantee of detection. Observe that, after one round, we can derive $\mathcal{P}_{fp}$, which is the probability that a randomly chosen peer in the malicious peers set is a legitimate peer. If $\mathcal{P}_{fp}$ is less than some predefined threshold $\mathcal{P}^*_{fp}$, we can terminate the algorithm. The rationale is that if the error decreases to a pre-defined threshold, we can terminate the detection procedure. Therefore, the overall detection algorithm can be described as follows.

Alg. B3: Overall Detection Algorithm

$t \leftarrow 0$;
do { $t \leftarrow t + 1$;
execute the corresponding detection algorithm for round t;}
while $(\mathcal{P}_{fp}(t) > \mathcal{P}^*_{fp})$
remove peers in $\mathcal{M}^i(t)$ from the neighbor list $\mathcal{N}^i$;

Improvement on the probability of false negative $\mathcal{P}_{fn}$: In the previous parts, we studied the general case of multiple malicious peers and each malicious peer may pretend to be a good peer in uploading valid sub-pieces. While we can guarantee that as long as the detection process is executed for sufficient number of rounds, peers in the set $\mathcal{M}^i(t)$ are all malicious, which means that $\mathcal{P}_{fp}$ converges to zero, but $\mathcal{P}_{fn}$ converges to one. Furthermore, if we execute the detection process for large number of rounds such that all peers in $\mathcal{M}^i(t)$ are malicious, then only a small number of malicious peers are in $\mathcal{M}^i(t)$. For example, for $\mathcal{P}_{fn}(t)$ computed in lemma 5, if we set $\alpha = \delta = p = 0.1$, $N = 50$ and $k = 5$, then when $t \geq 200$, only less than 30% of malicious peers are in the set $\mathcal{M}^i(t)$. Let us now consider how to improve this performance measure.

Observe that when the detection process runs for sufficient number of rounds, we can guarantee that peers in $\mathcal{M}^i(t)$ are all malicious. The deficiency is that only a small number of malicious peers remain in $\mathcal{M}^i(t)$. So one can remove those detected malicious peers from $\mathcal{M}^i(t)$ (i.e., blacklist them for further data exchange) and repeat the detection process again. If this process is repeated several times, one can be certain in removing all malicious peers from $\mathcal{N}^i$. In the example cited above, the probability of false negative is 70%, and we only need to repeat the detection process eight times to guarantee with high probability of detecting all malicious peers. The final detection algorithm is as follows.

Alg. B4: Improved Detection Algorithm

repeat{
do {
if (the computed hash value is the same as the hash code) {
with probability p : $\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1)$;
with probability $1-p$: $\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1) \cap \bar{U}^i(t)$;}

else: $\mathcal{M}^i(t) \leftarrow \mathcal{M}^i(t-1)$;}
while (peers in $\mathcal{M}^i(t)$ are not all malicious)
remove peers in $\mathcal{M}^i(t)$ from the neighbor list $\mathcal{N}^i$; }
until (all malicious peers in $\mathcal{N}^i$ are detected)

Estimating k, the number of malicious peers in the neighborhood: In previous sections, we present the detection algorithms for different scenarios and provide the analysis on different performance measures. In the analysis, we assume that parameter k, the number of malicious peers among peer i's neighborhoods, is known in advance. However, this assumption may not be realistic in practice because peer i has no idea whether its neighbor is malicious or not. Let us discuss how to relax this assumption.

To get the intuitive insight, let us first consider the case corresponding to algorithm B1, in which the probability of false negative is just zero. In this case, all malicious peers will remain in $\mathcal{M}^i(t)$, so when the detection process converges, i.e., the set $\mathcal{M}^i(t)$ cannot be shrunk any more, the number of malicious peers k must be equal to $|\mathcal{M}^i(t)|$. Based on this fact, if we set $k = |\mathcal{M}^i(t)|$ and use it to calculate the performance measures, e.g., the probability of false positive $\mathcal{P}_{fp}(t)$, it must be a good approximation when t is sufficiently large. We need to mention that, even if $\mathcal{P}_{fn} = 0$, $\mathcal{P}_{fp}(t)$ still cannot be computed directly through $\frac{|\mathcal{M}^i(t)|-k}{|\mathcal{M}^i(t)|}$, but it can be computed by the way presented in Lemma 5. Now, let us consider the scenario corresponding to algorithm B2. In this case, we can estimate the parameter k via the known information $|\mathcal{M}^i(t)|$. Different from the previous case, $\mathcal{P}_{fn}$ is not zero any more, so even if the detection process converges, k is not equal to $|\mathcal{M}^i(t)|$. However, the probability that a malicious peer remains in the malicious peers set $\mathcal{M}^i(t)$ is $1 - \mathcal{P}_{fn}(t)$. So, to compute the probability of false positive, the unknown parameter k can be estimated via $\frac{|\mathcal{M}^i(t)|}{1-\mathcal{P}_{fn}(t)}$. Also $\mathcal{P}_{fn}(t)$ can be first approximated by the formula presented in Lemma 5, i.e., $\mathcal{P}_{fn}(t) = 1 - \left[1 - \alpha\delta(1-p)(\alpha\delta + 1 - \alpha)^{k-1}\right]^t$, where k can be simply set to be $|\mathcal{M}^i(t)|$. The rationale is that since we are only interested on the limiting result (i.e., t is sufficiently large), it is a good approximation even if we set $k = |\mathcal{M}^i(t)|$ when we compute $\mathcal{P}_{fn}(t)$.

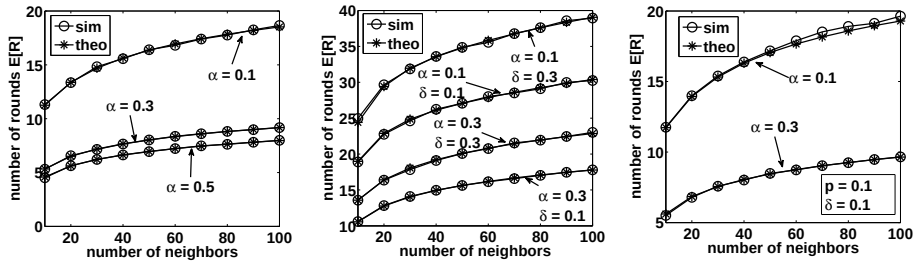
Based on the above arguments, we can measure the performance of the algorithms even if the parameter k is not known in advance. We summarize the results in lemma 6. The validation via simulation of this approximation is presented in Section 6.

Lemma 6. *If algorithm B1 is used , when t is sufficiently large, $\mathcal{P}_{fp}$ can be approximated as $\mathcal{P}_{fp}(t) = 1 - \frac{|\mathcal{M}^i(t)|}{|\mathcal{M}^i(t)|+(N-|\mathcal{M}^i(t)|)(1-\alpha(1-\alpha)^{|\mathcal{M}^i(t)|})^t}$. If algorithm B2 is used, when t is sufficiently large, $\mathcal{P}_{fn}$ can be approximated as $\mathcal{P}_{fn}(t) = 1 - \left[1 - \alpha\delta(1-p)(\alpha\delta + 1 - \alpha)^{|\mathcal{M}^i(t)|-1}\right]^t$, and $\mathcal{P}_{fp}$ can be approximated as $\mathcal{P}_{fp}(t) = 1 - \frac{k}{k+(N-k)(\frac{1-\alpha(1-p)(\alpha\delta+1-\alpha)^k}{1-\alpha\delta(1-p)(\alpha\delta+1-\alpha)^{k-1}})^t}$, where $k = \frac{|\mathcal{M}^i(t)|}{1-\mathcal{P}_{fn}(t)}$.* ∎

6. Simulation and Models Validation

In this section, we validate our models by comparing them with simulation results. We present the performance measures like $\mathcal{P}_{fp}$, $\mathcal{P}_{fn}$ and $E[\mathcal{R}]$. Since our detection algorithms are totally distributed, so in the simulation, we only focus on a particular legitimate node, and consider the communication between this legitimate peer and its neighbors. On the other hand, our detection algorithms are executed round by round, and we take the duration of receiving all sub-pieces that are needed to reconstruct a piece as one round. In each round, only some of the neighbors may perform the data upload. So in the simulation, we set that each neighbor is chosen to upload with probability α in each round. The setting for other parameters in each scenario is presented in the following subsections.

Experiment 1 (Single Attacker): We first consider the case of a single malicious peer in $\mathcal{N}^i$. For this case, except for the improved algorithm, the probability of false negative $\mathcal{P}_{fn} = 0$, while the probability of false positive $\mathcal{P}_{fp}$ is zero provided that set $\mathcal{M}^i(t)$ is a singleton. Therefore, we only need to show the average number of rounds needed to shrink the size of $\mathcal{M}^i(t)$ to singleton.



(a) imitation probability $\delta = 0$ (b) imitation probability $\delta > 0$ (c) $\delta > 0$ and $p > 0$

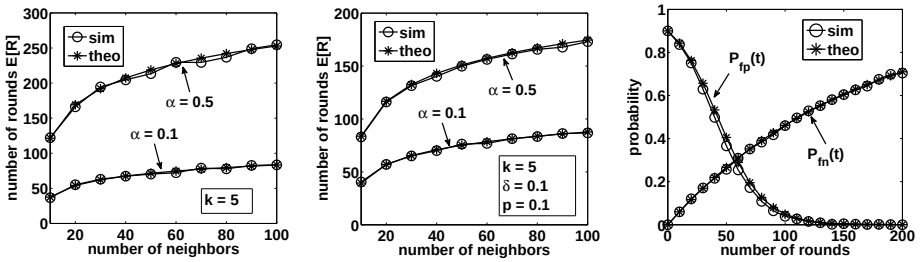
Figure 2: Average number of rounds ($E[\mathcal{R}]$) that peer i needs to detect the malicious peer vs. its neighborhood size ($|\mathcal{N}^i|$)

Fig. 2a corresponds to the case that the malicious peer does not pretend to be a good peer, or $\delta = 0$. So it always upload polluted sub-piece to peer i. We consider the random peer selection strategy in which each neighboring peer has the same probability α to be chosen to upload in each round. We vary α from 0.1 to 0.5 with a step size of 0.2. In this figure, the horizontal axis is the number of neighbors of peer i, or $|\mathcal{N}^i|$. The vertical axis is $E[\mathcal{R}]$, the average number of rounds for peer i to shrink $\mathcal{M}^i(t)$ to a singleton. We compare the simulation results with the theoretical results, which are computed via Eq. (2). First, we can see that the theoretical results fit very well with the simulation results. Secondly, when α increases, the number of rounds decreases. When $\alpha = 0.5$, the neighbors are almost evenly divided into two sets in each round, and the malicious peer only exists in one of them. So the set $\mathcal{M}^i(t)$ will converge faster

than other cases in which the neighboring peers are not evenly divided. The third observation is that peer i only needs less than 20 rounds of data exchange to discover the malicious peer, and this is indeed a welcoming result.

Fig. 2b corresponds to the case that the malicious peer may pretend to be a good peer in some rounds, or $\delta > 0$. In this figure, there are four simulation results and four theoretical results based on Eq. (3), and they correspond to different α and δ values ranging from 0.1 to 0.3. First, the theoretical results fit very well with the simulation results. Moreover, when α is fixed, $E[\mathcal{R}]$ increases as δ increases. This is because when δ gets larger, more rounds are needed to detect the malicious peer. Comparing Fig. 2b with Fig. 2a, we can see that more rounds are needed when the malicious peer pretends to be a good peer in some randomly chosen rounds. This also confirms our analytical results. In Fig. 2c, we illustrate the performance improvement (lower $E[\mathcal{R}]$) by using the randomized detection algorithm A3 by considering rounds in which the hash value matches as useful rounds with probability $1 - p$. We set $p = 0.1$ and see $E[\mathcal{R}]$ decreases comparing with the results shown in Fig. 2b.

Experiment 2 (Multiple Attackers): Let us consider the case of multiple malicious peers in $\mathcal{N}^i$. We first consider the scenario that malicious peers will not pretend to be a good peer, or $\delta = 0$. The probability of false negative is zero in this case. Since the malicious peers remain in the malicious peers set $\mathcal{M}^i(t)$, the probability of false positive $\mathcal{P}_{fp}(t)$ can be directly computed by definition, i.e., $\frac{|\mathcal{M}^i(t)|-k}{|\mathcal{M}^i(t)|}$. Fig. 3a shows the performance measure $E[\mathcal{R}]$ for both the theoretical results and simulation results. In Fig. 3a, the axes have the same



(a) average number of rounds for the case in which $\delta = 0$

(b) average number of rounds for the case in which $\delta > 0$

(c) probability of false positive and false negative for the case in which $\delta > 0$

Figure 3: multiple malicious peers

meanings as before. We set five malicious peers in $\mathcal{N}^i$. In this figure, there are two simulation curves and two theoretical curves corresponding to different α values from 0.1 to 0.5. First of all, we can see that the theoretical results based Eq. (6) fit well with the simulation results. Different from the single malicious peer case, when α gets larger, the number of rounds needed also increases. This is because when α gets larger, more peers are chosen to upload in each round,

then the hash code does not match with a higher probability. Since we discount those rounds that the hash code does not match in the baseline algorithm, therefore the average number of rounds needed to detect will increase.

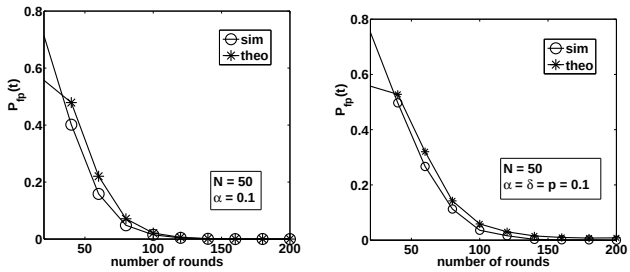
Now, consider the case that malicious peers can pretend to be good peers in some random rounds, or $\delta > 0$. There are five malicious peers in $\mathcal{N}^i$, and $\delta = 0.1$. Since the malicious peers may upload valid sub-pieces in some random rounds, even if the hash code matches, it is still possible that some malicious peers perform the upload. We use the proposed randomized detection algorithm and consider these as useful rounds with probability $1 - p$. In our simulation, we set $p = 0.1$. The simulation and theoretical results are shown in Fig. 3.

In Fig. 3b, the axes have the same meanings as before. There are two simulation curves and two theoretical curves corresponding to different α values from 0.1 to 0.5. Again, the theoretical results based on lemma 5 fit well with the simulation results. When α gets large, $E[\mathcal{R}]$ also increases. Fig. 3c shows the probability of false positive and false negative. There are 50 neighboring peers and five of them are malicious. We also set $\alpha = 0.1$. In this figure, the horizontal axis is the round number t and the vertical axis represents probabilities $\mathcal{P}_{fn}(t)$ and $\mathcal{P}_{fp}(t)$. From the figure, we can see that the $\mathcal{P}_{fp}$ quickly converges to zero as we increase the number of detection rounds, but $\mathcal{P}_{fn}(t) \approx 70\%$ when $t > 200$.

Experiment 3 (Performance of Alg. B4): Using the improved algorithm in Section 5, we can reduce $\mathcal{P}_{fn}$ by repeatedly running the detection algorithm. We carry out simulations to see the effect. In the simulation, we have $|\mathcal{N}^i| = 50$ and 10 of them are malicious. The parameters α, δ and p are all set to be 0.1. Since each time the malicious peer can only be detected with probability $1-\mathcal{P}_{fn}$, so we repeat the detection experiment three times (e.g., Exp. A, B and C) to see the performance of detecting all malicious peers. In all experiments, we only need to repeat the detection algorithm three times to detect all malicious peers. In the first experiment (Exp. A), 6 of the 10 malicious peers are detected in the first execution. In the second execution, 3 of the remaining 4 malicious peers are detected. In the final detection execution, the last malicious peer is detected. In the second experiment (Exp. B), the number of malicious peers that are detected in three executions are 8, 1 and 1 respectively. In the last experiment (Exp. C), the number of detected malicious peers in three executions becomes 4, 4 and 2. In summary, the improved detection algorithm is very efficient to detect all malicious peers.

Experiment 4 (Detecting Malicious Peers without Knowledge of k): Now, we consider the case that the parameter k is not known by the detecting peer. We validate the approximation analysis presented in lemma 6. In this simulation, the parameters α, δ and p are all set to be 0.1. We assume the detecting peer has 50 neighbors and five of them are malicious. However, the detecting peer does not know the number of malicious peers.

The simulation results and the theoretical results are shown in Fig. 4. We can see that when t becomes large, the theoretical results fit well with the simulation results. In other words, the analysis on the probability of false positive



(a) imitation probability $\delta = 0$ (b) imitation probability $\delta > 0$

Figure 4: The approximation analysis on the probability of false positive

presented in Lemma 6 is a *good* and *robust* approximation. Recall that, in the overall detection algorithm, the termination condition is designed based on the probability of false positive. Now we know that this probability can be well approximated even if the parameter k is not known in advance. Therefore, our detection algorithm is effective and can be easily implemented in practice.

7. Detection Optimization

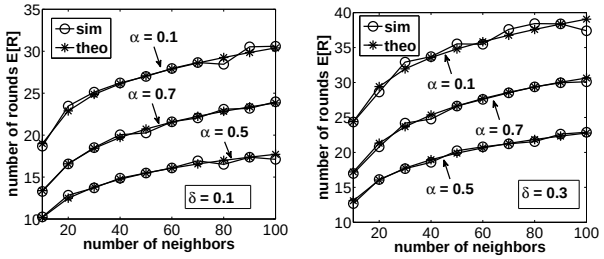
In this section, we explore how to utilize the performance models so that a good peer (e.g., peer i) can *quickly* detect malicious peers in its neighbors. Observe that, the key idea of the detection algorithm is to shrink the possible malicious peer set $\mathcal{M}^i(t)$ such that after some rounds, it only contains malicious peers. To decrease the number of rounds needed, we must shrink the set $\mathcal{M}^i(t)$ as much as possible in each round. As α is the main parameter that is determined by peer i, our goal is to determine how to choose α such that the number of total rounds is minimized. We define the shrinkage function $f(\alpha)$ to indicate the scale that the set $\mathcal{M}^i(t)$ being shrunk in each round, i.e., $|\mathcal{M}^i(t)| = f(\alpha)|\mathcal{M}^i(t-1)|$. In the following experiments, we use this shrinkage function to analyze several scenarios we studied in Section 4 and 5.

Scenario 1 (single malicious peer with $\delta > 0$) : In this scenario, peer i only has one malicious neighbor who may pretend to be a legitimate peer to upload correct sub-pieces with probability δ. Based on the detection algorithm A2, we can see that the potential malicious peer set $\mathcal{M}^i(t)$ will be shrunk only when the malicious peer uploads polluted sub-pieces, which occurs with probability $\alpha(1-\delta)$. Moreover, when the malicious peer set $\mathcal{M}^i(t)$ is shrunk, the shrink scale is α, i.e., $|\mathcal{M}^i(t+1)| = \alpha|\mathcal{M}^i(t)|$. The shrinkage function $f(\alpha)$ can be computed as:

$$f(\alpha) = \alpha(1-\delta)\cdot\alpha + (1-\alpha(1-\delta)) = (1-\delta)(\alpha - \frac{1}{2})^2 + 1 - \frac{1-\delta}{4}. \quad (8)$$

It is easy to show that $f(\alpha)$ is minimized at $\alpha^* = 0.5$.

***Simulation A*:** the simulation results are shown in Fig. 5. We consider different values of δ. In each figure, there are three groups of curves corresponding to α values 0.1, 0.5 and 0.7. We see that in both figures, $E[\mathcal{R}]$ is smallest when $\alpha = 0.5$. These results confirm with our analysis.



(a) $\delta = 0.1$ (b) $\delta = 0.3$

Figure 5: optimal α to minimize $E[\mathcal{R}]$: $\alpha^* = 0.5$

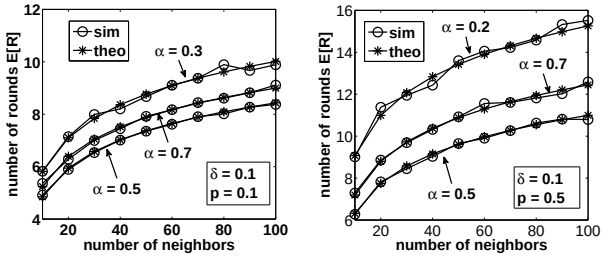
Let us consider the improved algorithm A3 in Section 4. In this case, the malicious peer set $\mathcal{M}^i(t)$ will be shrunk not only when the malicious peer uploads polluted sub-pieces, which occurs with probability $\alpha(1-\delta)$ and the shrink scale is α, but will also be shrunk when the malicious peer uploads correct sub-pieces, or it is not chosen to upload and at the same time this round is considered useful. This happens with probability $(1-\alpha(1-\delta))(1-p)$. The shrink scale is $1-\alpha$, i.e., $|\mathcal{M}^i(t+1)| = (1-\alpha)|\mathcal{M}^i(t)|$. The shrinkage function $f(\alpha)$ is computed as follows:

$$\begin{aligned} f(\alpha) &= \alpha(1-\delta)\cdot\alpha + (1-\alpha(1-\delta))(1-p)\cdot(1-\alpha) + (1-\alpha(1-\delta))\cdot p \\ &= (1-\delta)(2-p)(\alpha - \frac{1-p+1-\delta}{2(1-\delta)(2-p)})^2 + 1 - \frac{(1-p+1-\delta)^2}{4(1-\delta)(2-p)}. \end{aligned} \quad (9)$$

We can assume $p \geq \delta$, which implies $\frac{1-p+1-\delta}{2(1-\delta)(2-p)}$ is between $[0,1]$. So $f(\alpha)$ is minimized at $\alpha^* = (1-p+1-\delta)/(2(1-\delta)(2-p))$.

***Simulation B*:** the simulation results of this case are shown in Fig. 6. Fig. 6a shows the results when $\delta = p = 0.1$ and Fig. 6b shows the results when $\delta = 0.1$ and $p = 0.5$. From Eq. (9), we know that $\alpha^* \approx 0.526$ when $\delta = p = 0.1$ and $\alpha^* \approx 0.519$ when $p = 0.5$. In both figures, the horizontal axis is $|\mathcal{N}^i|$ and the vertical axis is $E[\mathcal{R}]$. In Fig. 6a, there are three groups of curves corresponding to α values 0.3, 0.5 and 0.7. We can see that $\alpha = 0.5$ is better than $\alpha = 0.3$ and $\alpha = 0.7$, which confirms our analysis. Similarly, in Fig. 6b, there are also three groups of curves corresponding to $\alpha = 0.2$, 0.5 and 0.7. The curve of $\alpha = 0.5$ shows the best results. Again, it validates our analysis.

Scenario 2 (multiple malicious peers with $\delta > 0$) : This case corresponds to the detection algorithm B2. In this case, the set $\mathcal{M}^i(t)$ will be shrunk only when the hash value matches and at the same time, this round is considered



(a) $\delta = p = 0.1$ so $\alpha^* \approx 0.526$ (b) $\delta = 0.1$ and $p = 0.5$ so $\alpha^* \approx 0.519$

Figure 6: optimal α to minimize $E[\mathcal{R}]$

useful by the randomized algorithm B2. This will happen with probability $(\alpha\delta + (1-\alpha))^k(1-p)$. If the set $\mathcal{M}^i(t)$ can be shrunk, the shrink scale is $1-\alpha$, i.e. $|\mathcal{M}^i(t+1)| = (1-\alpha)|\mathcal{M}^i(t)|$ and the function is:

$$\begin{aligned} f(\alpha) &= (\alpha\delta+1-\alpha)^k(1-p)\cdot(1-\alpha)+(1-(\alpha\delta+1-\alpha)^k(1-p)) \\ &= 1-\alpha(1-p)(\alpha\delta+1-\alpha)^k. \end{aligned} \tag{10}$$

The derivative of $f(\alpha)$ is:

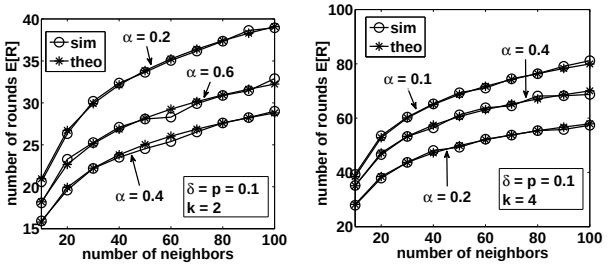
$$f'(\alpha) = (1-p)(\alpha\delta+1-\alpha)^{k-1}((k+1)(1-\delta)\alpha - 1).$$

So $f(\alpha)$ is minimized when $\alpha^* = 1/((k+1)(1-\delta))$.

***Simulation*:** simulation results are shown in Fig. 7. We set $\delta = p = 0.1$. In Fig. 7a, we set $k = 2$ and show three groups of curves which correspond to $\alpha = 0.2$, 0.4 and 0.6. We can see that $\alpha = 0.4$ shows the best results, which fits our analysis of $\alpha^* \approx 0.37$. In Fig. 7b, k is set to 4 and there are also three groups of curves. We observe that $\alpha = 0.2$ is better than $\alpha = 0.1$ and 0.4. Again, this confirms our analysis of $\alpha^* \approx 0.222$.

8. Related Work

P2P live streaming system has emerged and becomes very popular in recent years. Existing systems such as PPLive[1] and PPStream [2] use a data-driven overlay technique to provide video distribution to hundreds of thousands of concurrent users. In [8], authors address the design and implementation issues and also present large-scale P2P video streaming experiment. Modeling and measurements of P2P streaming are reported in [9, 10, 11]. Since P2P streaming system is decentralized, it is prone to various attacks. Recent security work mainly focused on the study of selfish behavior of peers and DoS attack[12, 13]. In [14], authors address the problem of dealing with cheaters in anonymous peer-to-peer networks. They present a protocol and show that it works well in theory and practice. The pollution attack in P2P file sharing system was first reported



(a) $\delta = p = 0.1$ and $k = 2$, so $\alpha^* \approx 0.37$

(b) $\delta = 0.1$, $p = 0.5$ and $k = 4$, so $\alpha^* \approx 0.222$

Figure 7: optimal α to minimize $E[\mathcal{R}]$

in [6]. In [5], experiments on real streaming system are presented and results show that pollution attacks can significantly reduce the number of legitimate peers and creates a devastating effect. Several various reputation systems are presented in [15, 16] for pollution filtering in file sharing system. The realtime implementation of a reputation system, Gredence, is proposed by Walsh and Sirer in [17, 16]. Related papers on misbehaving detection and analysis can be found in [18]. In our paper, we model the pollution attackers not as selfish peers, but as malicious peers. We provide a set of distributed algorithms to detect them. Moreover, we also give the rigorous analysis on the performance measures of our detection algorithms and validate them by extensive simulation.

9. Conclusion

In this paper, we present a set of *randomized* and *fully distributed* defense and detection algorithms to address the challenging issue of pollution attacks in P2P live streaming systems. Each peer can perform the detection. We consider both cases of (1) single attacker and (2) multiple attackers within the neighborhood of the detecting peer. We present the analytical methodology to evaluate such defense mechanisms and show how they can significantly improve the probability of false positive/negative, and how we can take advantage of the model so as to optimize the detection process to quickly discover attackers. Future work includes how to apply the detection algorithms and analytical methodology to other applications which requires nodes interactions, e.g., wireless mesh networks and online social networks.

Acknowledgement: this research is supported by the SHIAE 8115032.

References

[1] "http://www.pplive.com".

[2] "http://www.ppstreaming.com".

[3] "http://www.sopcast.com".

[4] Y. Huang, T. Z. Fu, D.-M. Chiu, J. C. Lui, C. Huang, "Challenges, Design and Analysis of a Large-scale P2P-VOD System", SIGCOMM Comput. Commun. Rev. (2008).

[5] P. Dhungel, X. Hei, K. W. Ross, N. Saxena, "The Pollution Attack in P2P Live Video Streaming: Measurement Results and Defenses", P2P-TV '07, 2007.

[6] J. Liang, R. Kumar, Y. Xi, K. Ross, "Pollution in P2P File Sharing Systems", INFOCOM, 2005.

[7] A. Shamir, "How to Share a Secret", CACM 22 (1979).

[8] B. Li, S. Xie, Y. Qu, G. Keung, C. Lin, J. Liu, X. Zhang, "Inside the New Coolstreaming: Principles, Measurements and Performance Implications", INFOCOM, 2008.

[9] Y. P. Zhou, D. M. Chiu, J. C. S. Lui, "A Simple Model for Analyzing P2P Streaming Protocols", ICNP, 2007.

[10] N. Parvez, C. L. Williamson, A. Mahanti, N. Carlsson, "Analysis of Bittorrent-like Protocols for On-demand Stored Media Streaming", SIGMETRICS, 2008.

[11] L. Ying, R. Srikant, S. Shakkottai, "The Asymptotic Behavior of Minimum Buffer Size Requirements in Large P2P Streaming Networks", in: http://arxiv.org/abs/0909.0763.

[12] W. Conner, K. Nahrstedt, I. Gupta, "Preventing DoS Attacks in Peer-to-Peer Media Streaming Systems", SPIE, 2006.

[13] W. Wang, Y. Xiong, Q. Zhang, S. Jamin, "Ripple-Stream: Safeguarding P2P Streaming Against DoS Attacks", ICME, 2006, pp. 1417 –1420.

[14] P. Gauthier, B. Bershad, S. Gribble, "Dealing with Cheaters in Anonymous Peer-to-Peer Networks", Technical Report 04-01-03, University of Washington, 2004.

[15] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, "The Eigentrust Algorithm for Reputation Management in P2P Networks", WWW, 2003.

[16] K. Walsh, E. G. Sirer, "Thwarting P2P Pollution using Object Reputation", Tech. Rep. CS Dept, Cornell university.

[17] K. Walsh, E. G. Sirer, "Fighting Peer-to-Peer Spam and Decoys with Object Reputation", P2PECON, New York, USA, 2005.

[18] G. Theodorakopoulos, J. Baras, "Malicious Users in Unstructured Networks", INFOCOM. 26th IEEE International Conference on Computer Communications. IEEE, 2007.