Peer-to-Peer Netw. Appl. (2017) 10:1079–1100
DOI 10.1007/s12083-016-0500-1

Protecting against malicious and selective forwarding attacks for P2P search & retrieval system

Yung-Ting Chuang[1]

Received: 1 March 2016 / Accepted: 31 July 2016 / Published online: 24 August 2016

Abstract With recent advances in networking technology, emerging networks continue to play an increasing role in the lives of most users. The Internet search and retrieval system is so powerful that it helps us to share information and perspectives from across the world. However, the threat of censorship exists on some centralized search engines, since all of their information is currently controlled by these sites administrators. The restriction and control of information are pervasive enough within governments and organizations to censor or intrude on even the most free and uncontrolled communication media. For this reason, the Peer-to-Peer (P2P) search and retrieval system is designed to resist censorship over the network. Nevertheless, its decentralized nature makes it very difficult to infer information that cannot be measured directly, such as the proportion of subverted and selfish nodes. Moreover, the situation is even more challenging when the network becomes extremely large. Hence, I propose a dynamic adaptive algorithm that can: 1) tackle the censorship and security issues; 2) determine the proportion of subverted and selfish nodes; 3) defend against malicious and selective forwarding attacks by appropriately adjusting the number of requests to ensure high match probability; 4) guarantee robustness and scalability even with different random networks and varied network sizes. In several experiments, I demonstrate that my algorithm can effectively and accurately estimate these metrics and manage the system, even when the network has a large proportion of malicious nodes, a large proportion of selfish nodes, or a mere partial view of network membership.

Keywords Probability Density Function (pdf) · Peer-to-Peer (P2P) search and retrieval · Message forwarding · Probabilistic analysis · Distributed systems · Random networks

1 Introduction

Traditionally, network technology was developed to enable the transfer of information between devices, and intentionally transferred information from the server to the clients. With recent advancements in networking technology, emerging networks continue to play an increasing role in the lives of most users. The Internet, which is designed to be uncontrolled and unbiased, has become the most popular information source in the post-industrial world. The Internet is so powerful that it helps us to share information and perspectives from across the world. Similarly, search engines are a major driver of this revolution, which aims for unbiased and uncontrolled information retrieval.

However, the threat of censorship appears to be one of the problems on these centralized search engines, since the information is controlled by these sites administrators. Currently, we all trust those administrators to remain benign and allow us to access, publish, and retrieve the information. However, history has shown that restriction and control of information are pervasive enough within governments

✉ Yung-Ting Chuang
ytchuang@mis.ccu.edu.tw

[1] Department of Information Management, National Chung Cheng University, 168 University Rd., Minhsiung Township, Chia-Yi, 62100, Taiwan

and organizations to censor or intrude on even the most free and uncontrolled communication media. For example, internet censorship is prevalent in China and its social media [9]. According to Chomhaill et al., even though censorship is contrary to the very idea of freedom of speech [4], it seems to be continuing to grow [9] in China.

Thus, I had previously presented the design of the trustworthy information publication and retrieval system [11], which is an unstructured P2P search system that takes a probabilistic approach and avoids information censorship over the network. However, the decentralized nature of this system makes it very difficult to deduce the information that cannot be measured directly, such as the proportion of subverted nodes that do not reply when they have matches. When we considered the question of malicious nodes, we specifically considered an attack scenario in which the network has a large number of malicious or Sybil nodes that behave normally most of the time, but do not match requests and return responses. For example, BitTorrent [59] might suffer a Sybil attack when a few malicious nodes stand in for a larger number of nodes and refuse to respond when they have matches, thus affecting overall download time and retrieval rate. These Sybil attacks also exist in Instant Messaging (IM) applications, which use Distributed Hash Tables (DHT) routing, where a few malicious nodes might insert a large number of fake nodes and act as normal nodes in the network, but drop queries or refuse to reply to queries even when they have matches. As a result, I had previously introduced a novel algorithm [10] in which every node maintains a full view of the membership, and uses direct communications to protect against the subverted nodes in a small network. With [10], I demonstrated that the network can maintain the same high probability of a match when some of the nodes are subverted as when all of the nodes are operational.

Traditionally, when a source node needs to distribute its message to multiple nodes, it multicasts its messages directly to all of these nodes. However, this way might not be a feasible method of communication, especially when in a large network. As a result, most systems prefer an alternative strategy called probabilistic forwarding, such that the source node first transmits the message to a set of nodes, and has each of these nodes forward the messages to another set of nodes, and so on. Probabilistic forwarding is an important solution to multicasting and gossiping, and has been largely applied to sensors in wireless sensor networks (WSNs), mobile ad-hoc networks (MANETs), and vehicular ad-hoc networks (VANETs) [7, 26, 29, 53], where there are enormous numbers of mobile nodes distributed over a large area. For example, WSNs are widely applied to military sensing, habitat monitoring, and military tracking, where their sensors are gathered together to sense and share information with other nodes. Similarly, VANETs are applied to traffic information systems [52], where sensors collect and process current traffic data in cooperation with others. The advantage of such a message-forwarding strategy is that it allows nodes to share the load of processing, buffering, and communication costs with others, rather than placing the entire load on the source node. Moser and Melliar-Smith [46, 47] hence perform a probabilistic analysis that finds the probability density function (pdf) for the number of distinct nodes reached in terms of forwarding fanouts, forwarding levels, and forwarding probability.

Nevertheless, the above works [10, 46, 47] would not be applicable to a scenario that contains all of the following cases: 1) when the network size is extremely large; 2) when the network uses a message-forwarding strategy; 3) when there exist subverted nodes that do not respond when they have matches; or 4) when there exist selfish nodes that do not forward messages. When the network size is extremely large, such as in the case of file-sharing networks like Gnutella [49] or BitTorrent [13], the situation is even more challenging, since many of the nodes cannot afford to maintain a full view of the membership due to limitations on memory space and bandwidth. In addition, when considering networks where nodes are allowed only for short-range communication (i.e., wireless sensor networks (WSNs), mobile ad-hoc networks (MANETs), and vehicular ad-hoc networks (VANETs)), direct communication would not be feasible for data distribution and retrieval; thus, the source or requesting nodes would need to first forward their messages to a set of nodes, and then have each of these nodes forward the messages to another set of nodes. Moreover, when the network suffers Sybil attacks [27, 59], malicious nodes refuse to respond when they have matches. Furthermore, when the network has selected forwarding attacks [1, 63], selfish nodes do not forward any metadata or request messages. The fact that there is no work applicable to a scenario that contains all four of the cases mentioned above, and thus, this is why I have developed a dynamic adaptive message forwarding algorithm in this paper.

My algorithm can accurately estimate the proportion of subverted and selfish nodes, adaptively adjust the number of requests according to various network sizes, and achieve high match probability even when the network has a large proportion of malicious and selfish nodes. The following summarizes the contributions of this paper:

1. My algorithm tackles censorship and security issues, such that it can assure Internet users that a small number of administrators cannot prevent them from distributing their messages and information, or from retrieving information from others.

2. My algorithm employs the probability density function (pdf), Exponential Weighted Moving Average (EWMA) method, and modified chi-squared test to determine the proportion of subverted and selfish nodes.
3. My algorithm defends against malicious and selective forwarding attacks by appropriately adjusting the number of requests to ensure high match probability, even when the network has large proportions of subverted and selfish nodes.
4. My algorithm guarantees robustness and scalability in different random networks (e.g., Erdos-Renyi, Watts-Strogatz, and Barabasi-Abert networks) and across varied network sizes.

We know that it is important to develop such an algorithm for P2P search and retrieval systems before it is needed, to ensure that it is available when it is needed. As with other questions of trust, we must create our defenses before a problem arises.

Section 2 of this paper discusses related work, and Section 3 describes how my system works. Section 4 describes the architecture of the P2P search and retrieval system, and Section 5 provides foundations for the system. Section 6 describes the message forwarding algorithm, and Section 7 presents an experimental evaluation. Lastly, Section 8 presents the conclusion and future work.

2 Related work

2.1 Peer-to-peer network

Mischke and Stiller [44], Risson and Moors [50], along with Tsoumakos and Roussopoulos [57], provide comparisons of distributed search methods for peer-to-peer networks. The structured approach [5, 28] requires the nodes to be organized in an overlay network based on distributed hash tables, trees, or rings, which is efficient, but is vulnerable to control by its administrators. The unstructured approach [12, 22, 35, 49, 54, 55, 61] is typically a gossip-based approach, which uses randomization, and requires nodes to find each other by exchanging messages over existing links. My system, therefore, uses the unstructured approach, which is less vulnerable to manipulation.

Gnutella [49], one of the first unstructured networks, uses flooding of requests to find information. Freenet [12] is more sophisticated and efficient than Gnutella because it learns from previous requests. In Freenet, nodes that successfully respond to requests receive more metadata and more requests; thus, it is easy for a group of untrustworthy nodes to gather together and collect the most searches into their group, making Freenet vulnerable to subversion.

Ferreira et al. [22] used random walks to replicate both queries and data in an unstructured network. BubbleStorm [54], a probabilistic system for unstructured peer-to-peer search, replicates queries and data, and combines random walks with flooding. Pub-2-Sub [55] is a publish-and-subscribe service for unstructured P2P networks of cooperative nodes, as it uses directed routing to distribute subscription and publication messages to other nodes.

Quasar [61] and OneSwarm [35] are concerned with trust like my system. Quasar is a probabilistic publish-or-subscribe system that uses local gradients of aggregated vectors (e.g. gravity wells), and then applies a rendezvous-less event routing infrastructure to route messages directly to the nearby group members. The authors state that privacy and scalability concerns make centralized systems undesirable to the users of social networks. On the other hand, OneSwarm [35] is a P2P data-sharing system that allows data to be shared either publicly or anonymously, using a combination of trusted and untrusted nodes. My system does not use a structured overlay, nor use the combination of trusted and untrusted nodes, and has a different trust objective than Quasar and OneSwarm.

2.2 Exponential weighted moving average and chi-squared statistics

Several network security researchers use the EWMA algorithm and the chi-squared statistics. [25] applies the chi-squared test to detect intrusions, and [51, 58] discusses how to appropriately determine the smoothing factor for detecting network anomalies based on a known window size. Similarly, [64, 66] presents anomaly-detection techniques for intrusion detection based on the EWMA method and the chi-squared test. My dynamic adaptive message forwarding algorithm does not fix a window size, but rather, I consider all of the results starting from the time that a node joins to the network until the time it leaves the network. In addition, my algorithm uses the EWMA method and the modified chi-squared test to determine the proportion of malicious nodes in a node's view of the membership. Furthermore, I make the smoothing factor tunable, so that the user can freely set it for his/her particular network environment.

Press et al. [48] use a modified chi-squared statistic to balance the weights of the buckets, in favor of comparing two datasets. Similarly, Belen [3] and Heckert [32] apply modified chi-squared statistics to determine the similarities between attribute couples of a dataset and a projected subset. Similarly, in this paper, my algorithm employs the modified chi-squared statistic for achieving high accuracy in estimating the proportion of non-malicious and non-selfish nodes in the network.

2.3 Message forwarding and probability density function

Probabilistic forwarding, an alternative and important solution to multi-cast routing and gossiping, has been largely applied for sensors and ad-hoc networks [7, 26, 29, 53]. Hedetniemi [33] presents an overview of gossiping and broadcasting in communication networks. Similarly, Farley [20] presents algorithms that require nodes to broadcast with a minimum number of links, and determines upper and lower bounds of broadcast messages [19]. Likewise, [8, 18] uses forwarding in delay-tolerant networks to reduce overall performance cost. In addition, Deering et al. [15] present extensions to two routing algorithms, distance-vector routing and link-state routing, to achieve greater efficiency when messages are forwarded in both inter-networks and extended LANs.

Moser and Melliar-Smith [46, 47] present a paper that finds: 1) the probability density function (pdf) for the number of distinct nodes reached, 2) the expected number of distinct nodes to which a message is forwarded within a fixed and small size of the network, 3) the probability density function (pdf) for the number of new nodes at a given forwarding level, and 4) the expected number of new nodes at a given forwarding level. My scalable and adaptive forwarding algorithm thus applies the probability density function (pdf) like [46, 47] to estimate the expected number of distinct nodes that would receive the metadata and requests. However, rather than just estimating the expected number of requests and metadata, my algorithm calculates the match probabilities in terms of the forwarding fanout, forwarding level, forwarding probability, number of nodes that report matches, proportion of subverted nodes, and proportion of selfish nodes. In addition, my algorithm can estimate the proportion of subverted and selfish nodes, and can appropriately determine the appropriate number of requests to achieve high match probabilities.

2.4 Detecting and defending against malicious attacks

Some prior work has been focused on trustworthiness, in particular for detecting and protecting against malicious attacks. Jesi et al. [36] identify malicious nodes in an overlay network based on gossiping, and place such nodes on a blacklist. They focus on hub attacks in which malicious nodes collude and divide the network by spreading false rumors. [38] proposes a techniques to cope with malicious behavior, as well as coping with nodes that do not respond in order to reduce their workloads [31]. Some approaches use a prediction model obtained by training decision trees over a sequence of normal data [40], or use neutral networks to obtain a model [24] to detect novel attacks. [34] uses a statistical method for ranking each sequence and determining how often the sequence should occur in normal traces or in intrusions. Nevertheless, my algorithm does not use gossiping, training decision trees, or neutral networks to detect novel attacks, but rather, my algorithm detects malicious attacks based on the probability density function (pdf), EWMA method, modified chi-squared test, and the number of matches received for a node's request.

Condie et al. [14] present a protocol for finding adaptive P2P topologies to protect against malicious peers that upload corrupt, inauthentic, or misnamed content. Peers improve the trustworthiness of the network by forming connections based on its past trust scores, disconnecting malicious peers, and moving them to the edge of the network. Morselli et al. [45] describe an adaptive replication protocol that uses random walks, and have a feedback mechanism that adjusts the number of replicas according to the mean search length, for determining sufficient replications. However, my algorithm does not disconnect malicious nodes, nor does it use random walks; rather, my algorithm increases the number of nodes to distribute requests in order to maintain high probability, even when the network contains a large proportion of malicious nodes.

2.5 Detecting and defending against selective forwarding attacks

There are various techniques to detect and defend against selective forwarding attacks. According to [63], malicious nodes might refuse to forward certain messages, or drop the messages passing through them, resulting in a failure to propagate the messages any further. As noted in [1], selective forwarding attacks are even harder to detect in WSN, since sensor nodes all have limited signals.

Deng et al. [16] propose a Secure Data Transmission (SDT) that uses watermark technology to detect selective forwarding attacks. [30] uses Lightweight Detection to generate the alert packet and detects selective forwarding attacks. Alajmi and Elleithy [1] demonstrate that their system obtains safe data transmission between nodes and effectively detect the selective forwarding attacks, and achieves reasonable design issues. Similarly, Yu and Xiao [62, 65] present a Lightweight Security Scheme (LWSS) to detect selective forwarding attacks in WSN, which uses multi-hop acknowledgement techniques to launch alarms by obtaining responses from intermediate nodes. Yu and Xiao further claim that their scheme is reliable because they randomly select some nodes as the checkpoint to send acknowledgement in order to detect the adversary nodes. However, my algorithm detects selective forwarding attacks by applying the probability density function (pdf), EWMA method, and modified chi-squared test, using the number of matches received for a node's request to estimate the proportion of non-selfish nodes in the network, and thus is quite different from the techniques as discussed above.

Mathur et al. [41] propose an algorithm that detects black-hole and selective forwarding attacks, defends against these attacks by employing cryptographic hashes for black-hole attacks, and applies a neighborhood watch and threshold-based analysis to correct selective forwarding attacks. In [23], the authors uses multipath routing scheme to defend against selective forwarding attacks in WSN, such that when a node detects the packet drop during the routing process, it will resend the packet to the alternative route. Karlof et al. [37] propose multi-path routing to defend against the selective forwarding attacks, in which they basically allow every packet to choose its next hop probability from a set of possible nodes, thus reducing the chances of having adversary nodes gain complete control of data flow. Likewise, Lee and Cho [39] propose a fuzzy-based reliable data-delivery method to defend against selective forwarding attacks, where the number of paths for packet delivery is based on the fuzzy method. However, for protecting against selective forwarding attacks, my algorithm increases the number of nodes that forward the requests to maintain high match probability, rather than using threshold-based analysis, multipath routing, or a fuzzy-based scheme.

2.6 Random networks

Some prior work has focused on random networks that might contain the following properties: 1) the network has a small-world effect; 2) the network follows a power-law distribution; 3) the network follows a binomial distribution. The first property means that most nodes are reachable from other nodes within a small number of hops, and this property can also apply to the existing social networks [43]. The second property means that the probabilities of the node degrees vary as a power-law. The third property indicates that degree of any node follows a binomial distribution.

In my experimental evaluation section, I have considered three different random networks, namely, Erdos-Renyi (ER) network, Watts-Strogatz (WS) network, and Barabasi-Albert (BA) network, in the context of my algorithm for a P2P search and retrieval system. Therefore, I briefly introduce these random networks in the below subsections.

2.6.1 Erdos-Renyi (ER) network

Erdos-Renyi (ER) network [17], a classic random network that contains small-world effect, forms a connection between two nodes with random probabilities. The ER network has the following properties: 1) every link I_{ij} has a probability of $p \in (0, 1)$ of being selected, 2) every link I_{ij} is independent from other links, 3) the total number of edges is $\frac{p \times n(n-1)}{2}$, 4) the degree of any node forms a binomial distribution, 5) the presence of the link I_{ij} is based on:

$$I_{ij} = \begin{cases} 1, & \text{with probability } p \\ 0, & \text{with porbability } 1 - p \end{cases} \tag{1}$$

2.6.2 Watts-Strogatz (WS) network

The Watts-Strogatz (WS) network [60], another classic random network with small-world effects, is formed after the Erdos-Renyi (ER) network. The author created this network to address the limitations of ER networks, where the aim is to avoid a low clustering coefficient. Therefore, the WS network is claimed to solve the low-clustering issue, while still maintaining the short average path lengths of the ER network. The WS network is constructed with the following rules: 1) network initially places nodes to one or multi-dimensional lattices (e.g., circle or grid), 2) network connects each node with its $maxN$ of nearest neighbors, 3) nodes add a few long-range links based on the re-wiring probability β, such that the overall network has a short average path lengths [56], 4) the re-wiring probability β that connects the random pair of nodes for each link I_{ij} in the network is based on:

$$I_{ij} = \begin{cases} 1, & \text{with probability } \beta \\ 0, & \text{with porbability } 1 - \beta \end{cases} \tag{2}$$

If the re-wiring probability β is high, the network might have more chances to establish long-range connections between two random nodes. On the other hand, if the re-wiring probability β is low, the network would look the same as the original lattices.

2.6.3 Barabasi-Albert (BA) network

Barabasi-Albert (BA) network [2], a scale-free network with preferential attaching to high-degree nodes, follows a power-law distribution. As a result, the BA network tends to form a rich-get-richer process. The BA network contains the following properties: 1) network begins with an initial connected network of $smallN$ nodes, 2) when a new node joins the network, it forms a connection to the existing $smallN$ nodes with a connecting probability p_i, which is proportional to the summation of links from these existing $smallN$ nodes, 3) the connection probability p_i that a new node is connected to node i is based on:

$$p_i = \frac{k_i}{\sum_j k_j} \tag{3}$$

where k_i is the degree of node i, and j is the number of existing nodes. From Eq. 3, we can see that the high-degree nodes tend to accumulate more links, while low-degree nodes acquire hardly any new links.

3 System model

The steps of my system are given below:

1. A new node initially called $getBootView()$ (Section 4.1) to obtains some members from the bootstrapping nodes selected at random, and joins the network. Once that node has successfully joined to the system, it can become a source node or a requesting node at any time.
2. When a node wishes to distribute its metadata, it becomes a source node. The source node first calls $alDistribution()$ (Section 5.6) to determine the appropriate forwarding fanouts a and forwarding levels l, based on the expected number of nodes to which the metadata should be distributed m. Finally, the source node calls the $distribute()$ (Section 4.2) to distribute and forward the metadata message to randomly chosen nodes in its view.
3. Similarly, when a node wishes to distribute its request, it becomes a requesting node. The requesting node first calls the $EWMA()$ (Section 5.3) to calculate the current non-normalized observed probability array O, and then applies $xfGet()$ (Section 5.7) (the method that contains $norm()$ (Section 5.4) and $modChiSq()$ (Section 5.5)) to estimate the proportion of subverted and selfish nodes in the network, based on its previous request results. Next, the requesting node calls the $alGet()$ (Section 5.8) (the method that contains $pdf()$ (Section 5.1) and $findMatchProbability()$ (Section 5.2)) to calculate the appropriate value of a forwarding fanouts and the l levels of message forwarding to which a request is distributed. Finally, the requesting node calls $request()$ (Section 4.3) to distribute and forward the request to randomly chosen nodes in its view.

Figure 1 shows the flow diagram of my system, including all the steps mentioned above.

4 Design of P2P search and retrieval system

The P2P Search and Retrieval system is a fully distributed system with no centralized mechanisms and no centralized control. The nodes that participate in the network constitute the membership of the network. The source nodes produce metadata that describes its information, and distribute this metadata, along with the URL of the information, to a subset of the participating nodes selected at random. The requesting nodes generate requests that contain search keywords, and distribute the requests to a subset of the participating nodes selected at random. Nodes that receive such a request compare the keywords from the request with the metadata they hold. If there is a match, the matching node returns the URL of the associated information to the requesting node. The requesting node can then use this URL to further retrieve the information from the source node. In my system, nodes do not aim to maintain an agreed membership, but rather are allowed have their own and limited local view of the membership.

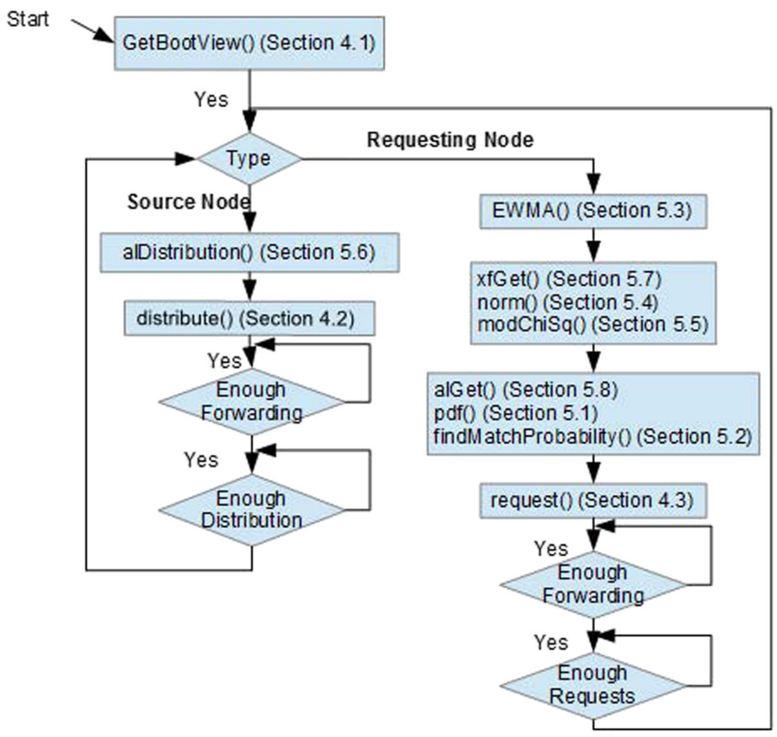

Fig. 1 System Flow Diagram

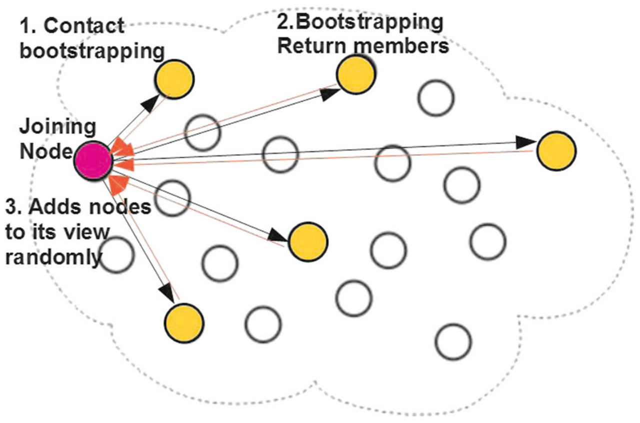
1. Contact bootstrapping
2.Bootstrapping Return members
Joining Node
3. Adds nodes to its view randomly

Fig. 2 A joining node first obtains membership from a set of bootstrapping nodes, randomly selects up to $maxN$ nodes, and adds them to its view

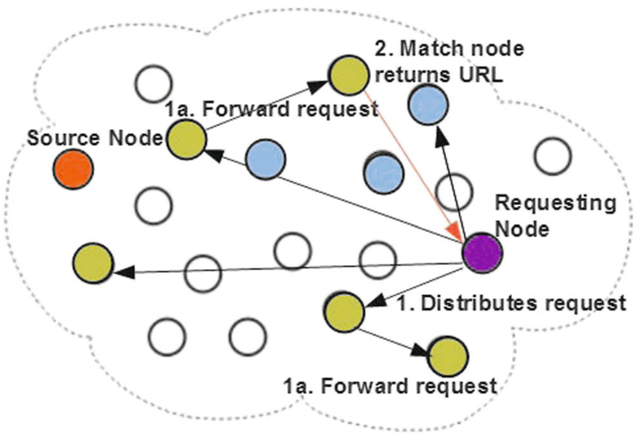

Fig. 4 The requesting node distributes requests. Some nodes might forward requests to other nodes. The matching node replies its URL

The following subsections describe the detailed design of my system.

4.1 Joining the membership, getBootView()

The steps of joining the membership are given below, and are shown in Fig. 2.

1. A node joining the membership first senses all of the nodes near itself. Next, it randomly selects a number of nodes to be its bootstrapping nodes, to which it needs to distribute membership requests, and asks for all of their members. Some of these bootstrapping nodes might forward these membership requests to other nodes in the network, based on the chosen forwarding probability.
2. The bootstrapping nodes return all of its members to the joining node.
3. The joining node then randomly chooses up to $maxN$ nodes, and adds them to its membership view.

4.2 Distribution metadata, distribute()

The steps involved when a source node distributes its metadata are given below, and are illustrated in Fig. 3

1. The source node first randomly chooses some nodes to which it needs to distribute its metadata. The source node then distributes its metadata together with the URL to the selected nodes.
2. Based on the chosen forwarding probability, some nodes might further forward this metadata to other nodes in the network.

4.3 Distribution requests, request()

The steps involved when a requesting node distributes requests are given below, and are illustrated in Figs. 4 and 5.

1. A requesting node randomly chooses some nodes to which it needs to distribute its requests, and distributes

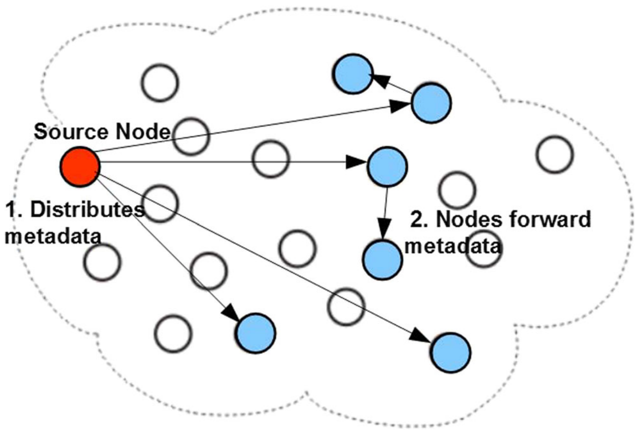

Fig. 3 A source node distributes its metadata and the URL of the advertisement to random nodes. Some nodes might forward metadata to other nodes

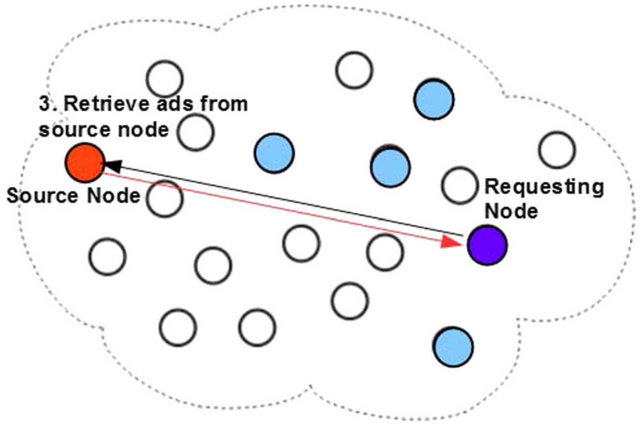

Fig. 5 The requesting node retrieves actual sources from the source node

its request to the selected nodes. Based on the chosen forwarding probability, some nodes might further forward this request to other nodes in the network.

2. A node that receives a request compares the keywords in the request with the metadata it holds. If it finds a match, the node responds to the requesting node with the URL.
3. The requesting node contacts and retrieves the actual sources from the source node.

5 Foundations

In this section, I describe the foundations of my proposed algorithm. The algorithm detects two kinds of attacks, where one attack is such that nodes do not match requests or metadata (called subverted nodes), and the other attack is such that nodes do not forward messages (called selfish nodes).

5.1 Probabilistic Analysis with probability density function, pdf()

In order to calculate the expected number of distinct nodes to which n_0's message is forwarded, I employ the following equation, as described in [6, 21, 46], to first calculate the probabilities of intersection between two subsets in order to eliminate duplicates. The parameters for the equation are described as follows:

- N: The set having cardinality n
- Y: A subset of N having cardinality y
- Z: A subset of N having cardinality z, where $y \leq z$.

The probability density function (pdf) $v(k)$ for $0 \leq k \leq y$, that $Y \cap Z$ has cardinality k is then given by:

$$v(k) = \binom{y}{k} \prod_{j=0}^{min(z,k-1)} \frac{z-j}{n-j} \prod_{j=0}^{min(n-z,y-k-1)} \frac{n-z-j}{n-k-j} \quad (4)$$

for $k-1 \geq 0$, $n-z \geq 0$, and $y-k-1 \geq 0$. If any of these conditions is not satisfied, then $v(k) = 1$.

Next, I employ the algorithm described in [46], to calculate the probability density function (pdf) [6, 21] for the number of nodes that receive the messages in terms of forwarding level, forwarding fanout, and forwarding probability. The algorithm calculates the probability density function (pdf) for the number of nodes at the level of message forwarding to which n_0's message is forwarded from levels 0 through L, and the probability density function (pdf) for the number of nodes to which n_0's message is forwarded from levels 0 through L. In addition, the algorithm calculates the expected number of nodes at level l, and the expected number of nodes from level 0 through L.

5.2 Match probabilities with message forwarding, findMatchProbability()

The primary parameters for the match probabilities with message forwarding are:

- a: The upper bound on the number of forwarding fanouts that a node should forward the message
- l: The upper bound on the number of forwarding levels that a node should forward the message
- m: The upper bound on the number of nodes in the network that have received n_0's metadata with fanout of a from level 0 to l, and is given by:

$$m = 1 + a + a^2 + \cdots + a^l = \begin{cases} \frac{a^{l+1}-1}{a-1} & \text{if } a > l \\ l+1 & \text{if } a = 1 \end{cases} \quad (5)$$

- r: The upper bound on the number of nodes in the network that have received n_0's requests with fanout of a from level 0 to l, and is given by:

$$r = 1 + a + a^2 + \cdots + a^l = \begin{cases} \frac{a^{l+1}-1}{a-1} & \text{if } a > l \\ l+1 & \text{if } a = 1 \end{cases} \quad (6)$$

- n: The number of participating nodes in the network.
- X: The number of participating nodes that are operational.
- k: The number of nodes that report matches to a requesting node.
- F: The number of participating nodes that are non-selfish and forward messages.

The analytical model is based on the hypergeometric distribution [21], which describes the number of successes in a sequence of random draws from a finite population without replacement. The analytical model that finds match probabilities of k matches is given by:

$$\begin{aligned} P(k) &= \frac{\binom{mx}{k}\binom{n-mx}{r-k}}{\binom{n}{r}} \\ &= \frac{(\frac{mx}{k}\frac{mx-1}{k-1}\cdots\frac{mx-k+1}{1})(\frac{n-mx}{r-k}\frac{n-mx-1}{r-k-1}\cdots\frac{n-mx-r+k+1}{1})}{(\frac{n}{r}\frac{n-1}{r-1}\cdots\frac{n-r+1}{1})} \end{aligned} \quad (7)$$

for $mx + r \leq n$ and $k \leq \min\{mx, r\}$. If either of those two conditions is not satisfied, then $P(k) = 0$.

From Eq. 7, the match probabilities $P(k \geq 1)$ of *one or more* matches is given by:

$$\begin{aligned} P(k \geq 1) &= 1 - P(0) \\ &= 1 - \frac{n-mx}{n}\frac{n-mx-1}{n-1}\cdots\frac{n-mx+1-r}{n-r+1} \end{aligned} \quad (8)$$

where $mx + r \leq n$.

The calculation for finding match probabilities of k matches with message forwarding is given in below, which combines both hypergeometric distribution and pdf() function [6, 21, 46, 47]:

$$MP = \sum_{m=a+1}^{n} \sum_{r=a+1}^{n} P(k) \times pdf(n, a, l, F).get(m) \times pdf(n, a, l, F).get(r) \quad (9)$$

Basically, the method calculates the probability of k match $P(k)$, as described in Eq. 7, and probabilistic analysis with probability density function (pdf), as described in Section 5.1 for both $pdf(n, a, l, F).get(m)$ and $pdf(n, a, l, F).get(r)$, for the number of nodes reached when forwarding to m and to r nodes. The match probability is the summation of the product of these match probabilities for all values of m and r.

5.3 Exponential Weighted Moving Average Algorithm, EWMA()

My dynamic adaptive forwarding algorithm uses the exponential weighted moving average (EWMA) algorithm to smooth the observations and average the non-normalized observations for k matches over a sequence of requests. In this way, the overall result would decrease the noise that is inherited in the individual samples.

The primary parameters for the EWMA algorithm are:

- c: The smoothing factor, $0 \leq c \leq 1$
- s_t: The output of the EWMA algorithm at time t.
- v_t: The current non-normalized observed probabilities $O(k)$ for k matches

The EWMA algorithm is defined by:

$$\begin{aligned} s_1 &= v_1 \\ s_t &= c \times v_t + (1 - c) \times s_{t-1} \quad \text{if } t > 1 \end{aligned} \quad (10)$$

In addition, I offer the user the opportunity to choose the appropriate value of c for his/her network environment. Different users, operating in different network environments with different objectives, might choose different values of c, and my system allows them to do so.

5.4 Normalization, norm()

In real-world deployments and in my experiments, it's impossible to use a request to estimate the proportion of subverted nodes and selfish nodes when all of the responses indicate no match. This is because such responses can arise in the following situations: 1) when the metadata and the requests are distributed to disjoint subsets of nodes, 2) when there exist subverted nodes that have a match and do not report it, 3) when there exist selfish nodes that have received a message but did not forward this message, and 4) when there exists no metadata that match a request.

Thus, I exclude requests for which all responses indicate no match. Moreover, for large values of k, the probability of k matches is negligibly small, so I further exclude those requests as well. That is, I first determine a value $kMax$, exclude requests that return k responses for $k > kMax$, and normalize the probabilities $P(k)$, $1 \leq k \leq kMax$. To find the normalized probabilities $Q(k)$, $1 \leq k \leq kMax$, I perform the following calculation:

$$Q(k) = \frac{P(k)}{\sum_{i=1}^{K} P(i)} \quad (11)$$

5.5 Modified chi-squared method, modChiSq()

My algorithm first uses the modified chi-squared goodness-of-fit test [48] to first compare the normalized observed probabilities with the normalized analytical probabilities for different values of X. Next, my algorithm determines which of the analytical curves best matches the observed probabilities, in order to estimate the proportion of non-subverted nodes in its membership view. The modified chi-squared statistic is given as follows:

$$\chi^2 = \sum_{k=1}^{kMax} \frac{(o_k - e_k)^2}{o_k + e_k} \quad (12)$$

where:

- o_k: The actual number of observations that fall into the kth bucket
- e_k: The expected number of observations for the kth bucket
- $kMax$: The number of buckets into which the observations fall

5.6 Calculate appropriate *a* and *l* for metadata distribution, alDistribution()

According to [42], Melliar-Smith et al. have shown that the probability of one or more matches satisfied:

$$P(k \geq 1) > 1 - \exp^{\frac{mrx}{n}} \quad (13)$$

If the source node distributes its metadata to $m = 2\sqrt{n}$ of nodes, and the requesting node distributes its requests to $r = 2\sqrt{n}$ of nodes, with $x = 1$ proportional of nodes that are operational, then

$$P(k \geq 1) > 1 - \exp^{\frac{2\sqrt{n}2\sqrt{n}}{n}} \geq 1 - \exp^{-4} > 0.9817 \quad (14)$$

Therefore, in order to obtain a high probability of one or more matches, I choose $m = 2\sqrt{n}$ for the total number of nodes to distribute metadata. However, since my algorithm follows a message forwarding approach, the source node would need to first determine an appropriate number of fanouts (a) and forwarding levels (l). Therefore, my algorithm applies the algorithm, as shown in Fig. 6, that determines the appropriate number of a and l, such that the overall number of distributions would still be appropriate to achieve $m = 2\sqrt{n}$ of nodes.

The parameters and variables for $alDistribution()$ that determine the appropriate value of a and l to which a source node distributes its metadata are described as follows:

- n: The number of nodes in a node's view of the membership.
- mL: The upper bound on the number of nodes in the network that should receive the source node's metadata.
- l: The upper bound on the number of forwarding levels that a node should forward the source node's metadata
- a: The upper bound on the number of forwarding fanouts that a node should forward the source node's metadata
- m: The expected number of nodes to which the metadata should be distributed

5.7 Detecting subverted and selfish nodes, xfGet()

If the requesting node determines that there exist nodes that receive both metadata and the requests, but did not report the match, those nodes are all considered subverted nodes. In addition, if the requesting node discovers that there exist nodes that did not forward the message, those nodes are considered selfish nodes. My algorithm for detecting subverted and selfish nodes estimates the proportion of non-subverted and non-selfish nodes. For given values of n, a, l, $kMax$, $xfGap$, and $xfMin$, my algorithm first computes the analytical probabilities for the number k of matches for various values of X and F. These values of X and F enable the algorithm to find the estimated proportion x of non-subverted nodes and the proportion of f of non-selfish nodes, which yields significant changes in the number of nodes to which to distribute requests.

My algorithm first collects data on the number of responses that a requesting node receives for its request using the probability density function (pdf), EWMA method, and modified chi-squared test. Next, it calculates the empirical probabilities from the data. The algorithm then applies the normalization method to exclude zero matches, since it cannot distinguish a case in which no metadata exists from a case in which no node holds both the metadata and the request, or in which nodes do not forward messages. Using the probability density function (pdf), EWMA method, and modified chi-squared test, my algorithm compares the normalized observed probabilities $O(k)$ and the normalized analytical probabilities $P[F, X](k)$, for $k = 1, 2, ..., kMax$ with the various combinations of curves $[F, X]$.

The total combinations of curves $[F, X]$ depend on the value of $xfGap$ and $xfMin$, and my system allows users to set these values for their particular network environments. When the value of $xfGap$ is low (e.g. $xfGap = 0.1$), it provides a faster reaction to change the values of x and f, as well as having a higher risk of making mistakes. On the other hand, when the value of $xfGap$ is high (e.g. $xfGap = 0.5$), it gives slower reactions to change the values of x and f, and has a lower risk of making wrong estimates. Consequently, for different network environments with different objectives, users might choose different value of $xfGap$, and my system allows them to do so. For example, if an user sets $xfGap = 0.3$, $xfMin = 0.4$, it will create the following 9 combinations of curves $[F, X]$: 1) $[F = 1, X = 1]$, 2) $[F = 1, X = 0.7]$, 3) $[F = 1, X = 0.4]$, 4) $[F = 0.7, X = 1]$, 5) $[F = 0.7, X = 0.7]$, 6) $[F = 0.7, X = 0.4]$, 7) $[F = 0.4, X = 1]$, 8) $[F = 0.4, X = 0.7]$, and 9) $[F = 0.4, X = 0.4]$.

After the system has finished creating the combination of curves $[F, X]$, my algorithm then chooses as the observed proportion of non-subverted and non-selfish nodes, the value of x and f for which the modified chi-squared value is the smallest. This value of f and x is the algorithm's best estimate of the proportion of non-subverted and non-selfish nodes, and corresponds to the curve with the best fit.

The parameters for detection method $xfGet()$ that determines the estimated proportion x of non-subverted nodes and f of non-selfish nodes are described as follows:

Fig. 6 Method for finding the appropriate value of a and l for the metadata distribution

```
alDistribution($n$)
1 $a = 0; l = 0;$
2   $m \leftarrow 2\sqrt{n}$
3     $for(l \leftarrow 2$ to $10000)$
4       $for(a \leftarrow 1$ to $10000)$
5         $mL \leftarrow \sum_{i=1}^{l} a^l$
6         if $(mL > m)$ break
7 return $a, l$
```

- O_i: The array of unnormalized observed probabilities at time i.
- n: The number of nodes in a node's view of the membership.
- $kMax$: The upper bound on the number k of responses to a request.
- $P[F, X]$: The array of analytical probabilities for a particular value of F and X.
- x: The estimated proportion of non-subverted nodes in a node's view
- f: The estimated proportion of non-selfish nodes in a network that forward messages.
- l: The upper bound on the number of forwarding levels that a node should forward the requesting node's requests
- a: The upper bound on the number of forwarding fanouts that a node should forward the requesting node's requests
- $xfGap$: The gap between each curve $[F, X]$
- $xfMin$: The minimum value for F and X curve.

Pseudocode for detection algorithm is given in Fig. 7.

5.8 Protecting against subverted and selfish nodes, alGet()

If the detection algorithm estimates that the proportion x of non-subverted nodes is less than $x = 1.0$, or the proportion f of non-selfish nodes is less than $f = 1.0$, my algorithm makes an adjustment in which it first determines the value of y_0 for the point on the $X = 1.0$ and $F = 1.0$ curve, and then computes the probability of one or more matches to obtain $y_0 = P(k \geq 1)$. Finally, the algorithm increases the number of l levels and a fanouts to which the requests are distributed, to achieve the same probability of a match as when $X = 1.0$ and $F = 1.0$.

The parameters and variables for the defensive method $alGet()$ that determines the number a fanout and l levels to which a requesting node distributes its request are described as follows:

- n: The number of nodes in a node's view of the membership.
- y_0: The probability $y_0 = P(k \geq 1)$ of one or more matches when $r = m = 2\sqrt{n}$ and $X = 1.0$.
- x: The estimated value of x returned by the detection method $xfGet()$.
- f: The estimated value of f returned by the detection method $xfGet()$.
- a: The number of forwarding fanouts
- l: The number of forwarding levels

Pseudocode for the defensive method is given in Fig. 8.

6 Message forwarding algorithm

6.1 Dynamic adaptive message forwarding algorithm, dynamicAdaptive()

In this section, I describe my dynamic adaptive message-forwarding algorithm, dynamicAdaptive(), which deals with attacks by subverted and selfish nodes. In addition, my algorithm addresses situations where every node cannot have knowledge of all the nodes in its membership, especially on a very large-scale network. In summary, the aims of my algorithm are to: 1) utilize the probability density function (pdf), EWMA method, and modified chi-squared test to detect subverted and selfish nodes; 2) defend against subverted and selfish nodes by appropriately adjusting the number of requests to achieve high match probability; and 3) ensure network robustness and scalability with different random networks and varied network sizes. The basic idea of my algorithm is described as follows, and the pseudocode for the dynamic adaptive algorithm is given in Fig. 9.

Fig. 7 Detecting values of subverted (x) and selfish (f) nodes

```
xfGet($O, n, a, l, kMax, xfGap, xfMin$)
1  $O \leftarrow$ norm($O, kMax$); $CC \leftarrow 0$
2  for ($i \leftarrow 1$ to $xfMin$; $i -= xfGap$) do
3    for ($j \leftarrow 1$ to $xfMin$; $j -= xfGap$) do
4      $[F, X][CC] \leftarrow i, j$
5      $CC{+}{+}$
6  for ($j \leftarrow 1$ to $CC$) do
7    $F, X \leftarrow [F, X][j]$
8    for ($KK \leftarrow 1$ to $kMax$) do
9      $P[F, X][KK] \leftarrow$ findMatchProbability($n, a, l, F, X, KK$)
10     $P[F, X][KK] \leftarrow$ norm($P[F, X][KK], kMax$)
11     $modChiSq[F, X][KK] \leftarrow$ modChiSq($O, P[F, X][KK], kMax$)
12 $f, x \leftarrow$ min($modChiSq[F,X][0], modChiSq[F,X][1], \ldots modChiSq[F,X][CC]$)
13 return $f$, $x$
```

Fig. 8 Determining the value of a and l that maintains the same probability of a match when some of the nodes are subverted as when none of the nodes is subverted

```
alGet($n$, $y_0$, $x$, $f$)
1 $for(l \leftarrow 2$ to 10000)
2   $for(a \leftarrow 1$ to 10000)
3     $y \leftarrow 1-$ findMatchProbability$(n, a, l, f, x, 0)$
4     if $(y > y_0)$ break
5 return $a, l$
```

1. A newly joining node calls $getBootView()$ to get its initial view of the membership from a bootstrapping node, randomly chooses up to $maxN$ nodes, and adds them into its initial view.
2. A source node waits until the current $time$ reaches $nextTime$, or the time to send its next metadata message. It then calls the $alDistribution()$ method to determine the appropriate a and l for the metadata distribution, and then calls the $distribute()$ method to distribute the metadata message to randomly chosen nodes in its view. Some nodes may forward the metadata they receive to other nodes in the network.
3. Similarly, a requesting node waits until the current $time$ reaches $nextTime$, or the time to send its next request message. The requesting node first checks whether its a and l have been previously determined. If not, it sets $X = 1$ and $F = 1$, then calls $alGet()$ to calculate the value of a and l.
4. After that, the requesting node calls the $request()$ method to distribute the request to randomly chosen nodes in its view. Some nodes may forward the requests they receive to other nodes in the network.
5. The requesting node further applies the EWMA() method to calculate the current non-normalized observed probability array O.
6. Next, the requesting node waits until it has collected up to d requests, and then calls the $xfGet()$ method to estimate the proportion of subverted and selfish nodes in the network.
7. The requesting node checks the number of successive estimates $hair$ by a modified chi-squared test that indicates the same changes in the value of x and f, determines whether it should accept the change and allows the algorithm to take action to change the value of a and l. If the number of successive estimates is greater than $hair$, the requesting node calls the $alGet()$ method to adjust the number a forwarding fanouts and the l levels of message forwarding to which a request is distributed. Otherwise, the requesting node simply sets x and f to its previous determined values ($x = xPrev$, $f = fPrev$).
8. The requesting node sets the the previous x and previous f to its current values ($xPrev = x$, $fPrev = f$)
9. The algorithm then goes back to Line 3 and repeats these steps indefinitely.

Fig. 9 Pseudocode for the Dynamic Adaptive Message Forwarding Algorithm

```
Adaptive($n, kMax, c, d, maxN, hair, xfGap, xfMin$)
1  $x, xPrev, f, fPrev \leftarrow 1.0$; $i, numMatches, O(k) \leftarrow 0$
2  $view$ = getBootView($maxN$)
3  while (true) do
4    $i \leftarrow i + 1$
5    if ($isSourceNode$) then
6      $a, l \leftarrow$ alDistribute($n$)
7      distribute($view, numNodes, a, l$)
8    else if ($isRequestingNode$) then
9      if (firstTime)
10       $a, l \leftarrow$ alGet($n, 0.999, 1.0, 1.0$)
11     $numMatches$ = request($view, a, l$)
12     $prevO \leftarrow O$
13     $O \leftarrow$ EWMA($numMatches, prevO, kMax, c$)
14     if ($i >= d$)
15       $x, f \leftarrow$ xfGet($O, n, a, l, kMax, xfGap, xfMin$)
16     if (successive estimate of $x$ & $f \leq hair$)
17       $x \leftarrow xPrev$
18       $f \leftarrow fPrev$
19     else
20       $a, l \leftarrow$ alGet($n, 0.999, x, f$)
21     $xPrev \leftarrow x$
22     $fPrev \leftarrow f$
```

6.2 Non-adaptive message forwarding algorithm, nonAdaptive()

In this section, I describe my non-adaptive message-forwarding algorithm that deals with attacks by subverted and selfish nodes, nonAdaptive(). Basically, the non-adaptive algorithm is exactly the same as the dynamicAdaptive() algorithm, described in Section 6.1, except that it does not call the $alGet()$ method to adjust the value of a and l. In other words, the non-adaptive algorithm only focuses on detecting the proportion of subverted and selfish nodes in the network, but does not defend against these malicious and selfish nodes. The pseudocode for the non-adaptive algorithm is nearly the same as Fig. 9, except that lines 19, and 20 are omitted. The basic ideas of the non-adaptive algorithm and pseudocode are omitted due to space constraints.

7 Experimental evaluation

In this section, I performed a simulation to evaluate my non-adaptive and dynamic adaptive message-forwarding algorithms. My system operates over the Internet, and is implemented using HTTP, which operates over TCP. As such, communication is "reliable". However, nodes may become malicious and selfish at any time. That is, nodes in the network are assumed to follow the algorithms, except that:

– A node is allowed to have a certain number of members in its local view, in which case it only maintains a partial view of the actual network.
– A malicious node responds to a request but, if it has a match between the keywords in a request and the metadata it holds, it fails to report that match.
– A selfish node that sends metadata responds to requests, but does not forward messages.

In the simulation program, I first initialized the numbers n, $maxN$, $kMax$, c, d, a, l, and $hair$. Next, I constructed a random network (e.g., an Erdos-Renyi network, a Watts-Strogatz network, or a Barabasi-Albert network), and added up to $maxN$ of nodes to each node's view. Then, at each time step, some nodes might distribute metadata messages, distribute request messages, or forward messages. Some nodes might behave maliciously by not responding to a request with a match when they do have a match. Similarly, some nodes might behave selfishly by not forwarding to a message. Then, I evaluated the network robustness and scalability for the following scenarios: 1) various network sizes, 2) various values of $maxN$, 3) various random networks, 4) various values of X, and 5) various values of F. Lastly, in order to reflect a realistic network, I designed an extended scenario such that X and F increases or decreases its value slowly, and tested the effectiveness and scalability of my adaptive algorithm with this extended scenario.

7.1 Controlled variables

Malicious nodes can disrupt the behavior of the system and network by hiding or censoring information. Thus, my adaptive algorithm must adapt to circumvent such malicious nodes. A node can't know or control proportion of the malicious nodes in the actual membership, but it can adjust the number of request messages in order to obtain more responses, and obtain high match probability. To do so, nodes make use of the following quantity:

– X: The proportion of non-malicious nodes in the actual membership at a particular point in time.

When the overall network size is extremely large (e.g., $N >> maxN$), it is unrealistic and difficult to have all nodes maintaining a full view of the membership due to limitations on memory space and bandwidth. In other words, when a node's maximum view $maxN$ is much smaller than the actual network size N, it would contain too many undiscovered nodes from the network. As a result, my adaptive algorithm must adapt to circumvent such a scenario, and still allow nodes to obtain high match probabilities even when $maxN$ is much lower than N. For my algorithm, a node cannot directly detect the number of nodes that it has not discovered, but it can adjust the number of nodes to which it distributes its requests in order to obtain more responses to its requests. In my system, each node has the following quantity:

– $maxN$: This is the maximum number of nodes that a node is allowed to have in its current view.

Similarly, selfish nodes can disrupt the behavior of the system and network by ceasing the information flow to other nodes. As a result, my adaptive algorithm must adapt to circumstances when selfish nodes send metadata or respond to requests, but do not forward messages. A node cannot control the proportion of selfish nodes in the network, but it can adjust the number of forwarding levels and the number of forwarding messages to obtain more responses to its requests and ensure high match probability. In order to do so, nodes make the use of the following quantity:

– F: The probability of message forwarding in the network at a particular point in time, where $0 < F \leq 1.0$.

7.2 Performance metrics

The performance metrics for the scalable and dynamic message-forwarding algorithm are explained as follows:

- MP: The match probability of one or more responses for a request, averaged over all requesting nodes.
- MC: The message cost per node per time unit.
- x: The estimated proportion of non-malicious nodes in the actual network at a particular point in time.
- f: The estimated probability of non-selfish nodes in the actual network at a particular point of time.
- MA: The accuracy of my algorithm for estimating the correct values of f and x.

7.3 Varying *n* and *a* with fixed *maxN* on Erdos-Renyi network

I first considered the scenario in which I applied a non-adaptive algorithm and varied n and a with fixed $maxN$ on an Erdos-Renyi network. In the simulation, I set $maxN = 1000$, $n = 1000, 5000, 10000$, $a = 8, 12, 14$, $kMax = 9$, $c = 0.999$, $d = 50$, $l = 2$, $hair = 6$, $xfGap = 0.3$, and $xfMin = 0.4$. I performed the experiment on the following nine cases:

- Case 1: $F = 1.0$, $X = 1.0$
- Case 2: $F = 1.0$, $X = 0.7$
- Case 3: $F = 1.0$, $X = 0.4$
- Case 4: $F = 0.7$, $X = 1.0$
- Case 5: $F = 0.7$, $X = 0.7$
- Case 6: $F = 0.7$, $X = 0.4$
- Case 7: $F = 0.4$, $X = 1.0$
- Case 8: $F = 0.4$, $X = 0.7$
- Case 9: $F = 0.4$, $X = 0.4$

Figure 10 shows the MP curves of: 1) $n = 1000$, $a = 8$; 2) $n = 5000$, $a = 12$; and 3) $n = 10000$, $a = 14$, where all the curves were generated with a non-adaptive algorithm on an Erdos-Renyi network. Each node is allowed to have its current view of the membership up to $maxN = 1000$, regardless of actual network size n. In Fig. 10, I first noticed that when X and F decreases, the MP also decreases for $n = 1000$, $n = 5000$, and $n = 10000$. Similarly, when both X and F decrease, the MP dramatically drops to about $MP = 0.301818$ for $n = 1000$, $n = 5000$, and $n = 10000$. In addition, for $n = 1000$, $n = 5000$, and $n = 10000$, the mean value of MC is $MC = 72$ for $n = 1000$, $MC = 156$ for $n = 5000$, and $MC = 210$ for $n = 10000$. Lastly, I noticed that the match probabilities MP for $n = 1000$ are all very close to $n = 5000$ and $n = 10000$. This shows the effectiveness of my algorithm, because a node can still obtains similar match probabilities, even with different values of n. Subsequently, I chose $n = 10000$, $a = 14$, for the following experiment.

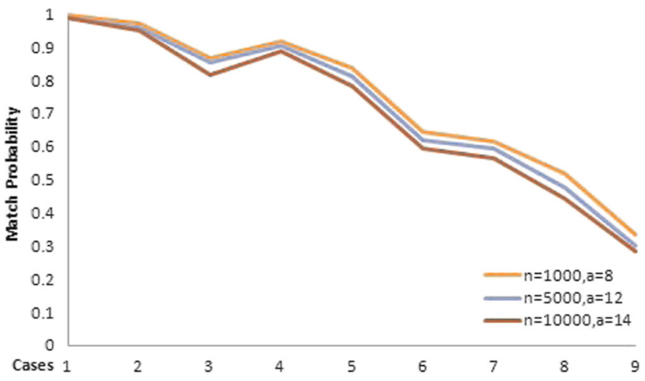

Fig. 10 MP where $n = 1000, 5000, 10000$, $a = 8, 12, 14$, $l = 2$, $maxN = 1000$, $kMax = 9$, $c = 0.999$, $d = 50$, $hair = 6$, $xfGap = 0.3$, $xfMin = 0.4$ with non-adaptive algorithm on Erdos-Renyi network

7.4 Varying *maxN* with fixed *n* on Erdos-Renyi network

I considered a scenario in which I applied a non-adaptive algorithm and varied $maxN$ with fixed n on an Erdos-Renyi network. In the simulation, I set $n = 10000$, $maxN = 1000, 5000, 10000$, $a = 14$, $kMax = 9$, $c = 0.999$, $d = 50$, $l = 2$, $hair = 6$, $xfGap = 0.3$, and $xfMin = 0.4$. I performed the experiment on the following nine cases:

- Case 1: $F = 1.0$, $X = 1.0$
- Case 2: $F = 1.0$, $X = 0.7$
- Case 3: $F = 1.0$, $X = 0.4$
- Case 4: $F = 0.7$, $X = 1.0$
- Case 5: $F = 0.7$, $X = 0.7$
- Case 6: $F = 0.7$, $X = 0.4$
- Case 7: $F = 0.4$, $X = 1.0$
- Case 8: $F = 0.4$, $X = 0.7$
- Case 9: $F = 0.4$, $X = 0.4$

Figure 11 shows the MP curves of $maxN = 1000$, $maxN = 5000$, and $maxN = 10000$, where all the curves were generated with a non-adaptive algorithm on an

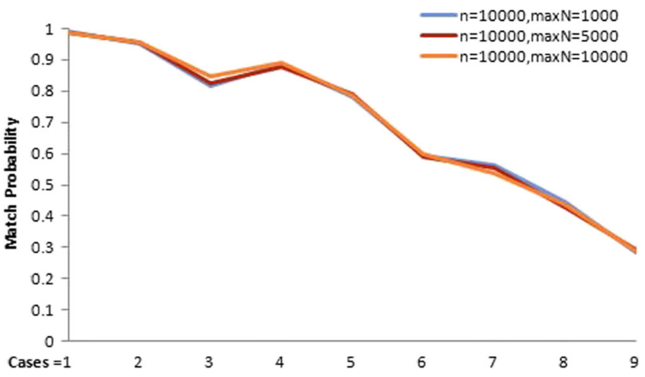

Fig. 11 MP where $maxN = 1000, 5000, 10000$, $a = 14$, $l = 2$, $n = 10000$, $kMax = 9$, $c = 0.999$, $d = 50$, $hair = 6$, $xfGap = 0.3$, $xfMin = 0.4$ with non-adaptive algorithm on Erdos-Renyi network

Erdos-Renyi network. In Fig. 11, I first noticed that when X and F decrease, the MP also decreases for $maxN = 1000$, $maxN = 5000$, and $maxN = 10000$. Similarly, when both X and F decrease, the MP dramatically drops to about $MP = 0.294318$ for $maxN = 1000$, $maxN = 5000$, and $maxN = 10000$. In addition, for $maxN = 1000$, $maxN = 5000$, and $maxN = 10000$, the overall mean values of MC are all $MC = 210$. Moreover, I noticed that the match probability for $maxN = 1000$ is close to $maxN = 5000$ and $maxN = 10000$. Lastly, by comparing Figs. 10 and 11, I concluded that even when the network has different values of $maxN$ or n, my algorithm still produces a similar MP. Consequently, I chose $maxN = 1000$ for my following experiment.

7.5 Varying different random networks

In this section, I considered a scenario in which I applied a non-adaptive algorithm and set $n = 10000$, $a = 14$, $l = 2$, $maxN = 1000$, $kMax = 9$, $c = 0.999$, $d = 50$, $hair = 6$, $xfGap = 0.3$, and $xfMin = 0.4$ on an ER network, a WS network, and a BA network. For the WS and BA networks, the source node distributes its metadata evenly across the overall network in order to obtain a fair amount of responses to its request messages. In addition, by having metadata distribute evenly to the network, some popular nodes would not be more vulnerable to attacks. I performed the experiment on the following nine cases:

- Case 1: $F = 1.0$, $X = 1.0$
- Case 2: $F = 1.0$, $X = 0.7$
- Case 3: $F = 1.0$, $X = 0.4$
- Case 4: $F = 0.7$, $X = 1.0$
- Case 5: $F = 0.7$, $X = 0.7$
- Case 6: $F = 0.7$, $X = 0.4$
- Case 7: $F = 0.4$, $X = 1.0$
- Case 8: $F = 0.4$, $X = 0.7$
- Case 9: $F = 0.4$, $X = 0.4$

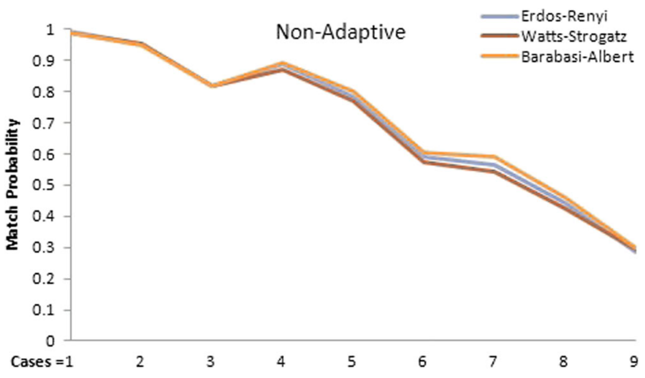

Fig. 12 MP with non-adaptive algorithm on ER network, WS network, and BA network, where $n = 10000$, $a = 14$, $l = 2$, $maxN = 1000$, $kMax = 9$, $c = 0.999$, $d = 50$, $hair = 6$, $xfGap = 0.3$, $xfMin = 0.4$

Figure 12 shows the MP curves with non-adaptive algorithms on the ER network, WS network, and BA network. In Fig. 12, I first noticed that when X and F decrease, the MP also decreases for all three random network. Similarly, when both X and F decrease, the MP dramatically decreases to about $MP = 0.292045$ for all three random networks. In addition, for all three random networks, the overall mean values of MC are all $MC = 210$. Moreover, I noticed that the match probability for the ER network remains quite close to that for the WS network and BA network.

Figure 13 shows the MA curves with a non-adaptive algorithm on the ER network, WS network, and BA network. In Fig. 13, I noticed that the accuracy MA for detecting the value of X and F remains high on all three random networks. As a result, these figures demonstrate the robustness and scalability of my algorithm, because a node can still obtain similar match probabilities with different random networks.

7.6 Non-adaptive vs. dynamic adaptive algorithm

In this section, I considered a scenario where I compared the non-adaptive and dynamic adaptive message-forwarding algorithm on the ER network, WS network, and BA network. In the simulation, I set $n = 10000$, $a = 14$, $l = 2$, $maxN = 1000$, $kMax = 9$, $c = 0.999$, $d = 50$, $hair = 6$, $xfGap = 0.3$, $xfMin = 0.4$, and performed the experiment using the following nine cases:

- Case 1: $F = 1.0$, $X = 1.0$
- Case 2: $F = 1.0$, $X = 0.7$
- Case 3: $F = 1.0$, $X = 0.4$
- Case 4: $F = 0.7$, $X = 1.0$
- Case 5: $F = 0.7$, $X = 0.7$
- Case 6: $F = 0.7$, $X = 0.4$
- Case 7: $F = 0.4$, $X = 1.0$
- Case 8: $F = 0.4$, $X = 0.7$
- Case 9: $F = 0.4$, $X = 0.4$

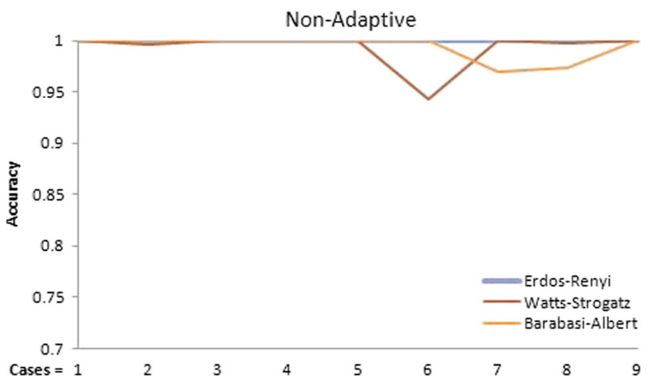

Fig. 13 MA with non-adaptive algorithm on ER network, WS network, and BA network, where $n = 10000$, $a = 14$, $l = 2$, $maxN = 1000$, $kMax = 9$, $c = 0.999$, $d = 50$, $hair = 6$, $xfGap = 0.3$, $xfMin = 0.4$

Table 1 MC for ER network

	Non-Adaptive	Adaptive
1	210	210
2	210	280.40
3	210	452.43
4	210	361.54
5	210	494.02
6	210	656.65
7	210	703.72
8	210	801.71
9	210	1105.10

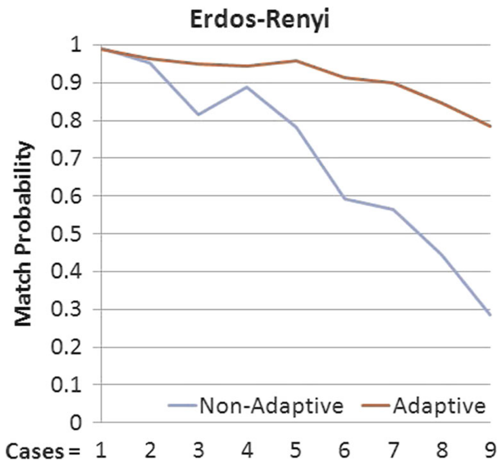

Fig. 14 MP for ER network

Table 2 MC for WS network

	Non-Adaptive	Adaptive
1	210	210
2	210	295.34
3	210	472.37
4	210	369.50
5	210	511.52
6	210	463.32
7	210	739.40
8	210	908.19
9	210	1145.06

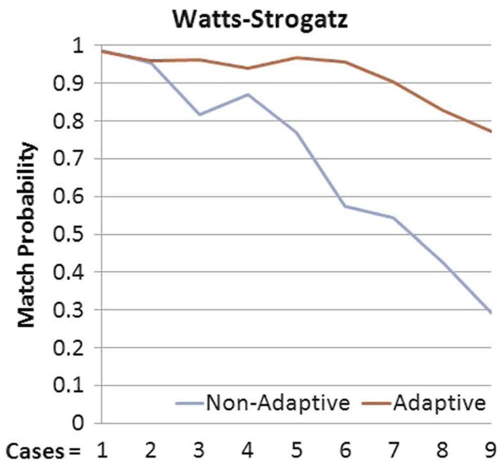

Fig. 15 MP for WS network

Table 3 MC for BA network

	Non-Adaptive	Adaptive
1	210	210
2	210	296.82
3	210	464.98
4	210	356.21
5	210	501.46
6	210	672.44
7	210	707.26
8	210	801.55
9	210	1156.53

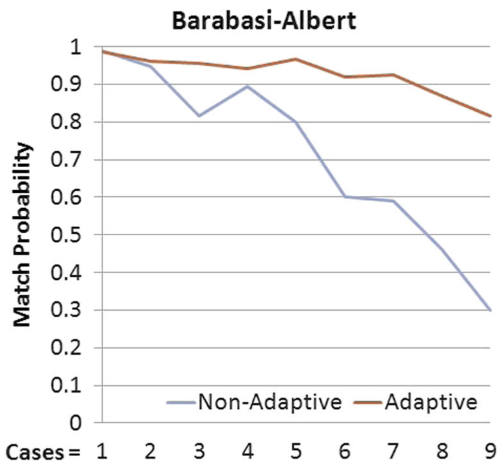

Fig. 16 MP for BA network

Tables 1, 2, and 3 show the message costs MC for non-adaptive and adaptive algorithms on the ER network, WS network, and BA network. First of all, the MC of the non-adaptive algorithm remains $MC = 210$ for all three networks. Next, as X and F decrease, MC from the adaptive algorithm increases. When the values of X and F are both low, such as $X = 0.4$ and $F = 0.4$, the MC increases moderately from $MC = 210$ to $MC = 1145.06$. The increased value of MC is still within a reasonable range because when X and F are both low (e.g. $X = 0.4$ and $F = 0.4$), the requesting node only increases its requests to about 10 % of the total nodes in the network. Lastly, I noticed that the message costs of non-adaptive and adaptive algorithms for the ER network are quite close to those of the WS network and BA network.

Figures 14, 15, and 16 show the MP for non-adaptive and adaptive algorithms on the ER network, WS network, and BA network. From these three figures, I noticed that when X and F decrease, the MP of the non-adaptive algorithm decreases for all three random networks. However, the MP of my adaptive algorithm increases significantly for all three random networks. In addition, these figures demonstrated the robustness and effectiveness of my dynamic adaptive algorithm, because when the network only has relatively low values of X and F, my algorithm can still adjust the number of requests to ensure high MP for all three random networks. For example, for $x = 0.4$ and $f = 0.4$, the MP is still about $MP = 0.81$ for all three random networks. Moreover, I observed that the MP of non-adaptive and adaptive algorithms for the ER network remains very close to the WS network and BA network.

7.7 Effectiveness of dynamic adaptive message forwarding algorithm

In this section, I considered an extended scenario in which I applied the dynamic adaptive algorithm and varied X and F with $n = 10000$, $a = 14$, $l = 2$, $maxN = 1000$, $kMax = 9$, $c = 0.999$, $d = 50$, $hair = 6$, $xfGap = 0.3$, $xfMin = 0.4$ on the ER network, WS network, and BA network. In order to reflect a realistic network, I designed my extended scenario such that X and F decrease or increase their value slowly, which comprises the following scenarios:

- Scenario 1: $X = 1$, $F = 1$, time $0 \sim 500$
- Scenario 2: $X = 0.95$, $F = 1$, time $500 \sim 600$
- Scenario 3: $X = 0.9$, $F = 1$, time $600 \sim 700$
- Scenario 4: $X = 0.85$, $F = 1$, time $700 \sim 800$
- Scenario 5: $X = 0.8$, $F = 1$, time $800 \sim 900$
- Scenario 6: $X = 0.75$, $F = 1$, time $900 \sim 1000$
- Scenario 7: $X = 0.7$, $F = 1$, time $1000 \sim 1500$
- Scenario 8: $X = 0.65$, $F = 1$, time $1500 \sim 1600$
- Scenario 9: $X = 0.6$, $F = 1$, time $1600 \sim 1700$
- Scenario 10: $X = 0.55$, $F = 1$, time $1700 \sim 1800$
- Scenario 11: $X = 0.5$, $F = 1$, time $1800 \sim 1900$
- Scenario 12: $X = 0.45$, $F = 1$, time $1900 \sim 2000$
- Scenario 13: $X = 0.4$, $F = 1$, time $2000 \sim 2500$
- Scenario 14: $X = 0.4$, $F = 0.95$, time $2500 \sim 2600$
- Scenario 15: $X = 0.4$, $F = 0.9$, time $2600 \sim 2700$
- Scenario 16: $X = 0.4$, $F = 0.85$, time $2700 \sim 2800$
- Scenario 17: $X = 0.4$, $F = 0.8$, time $2800 \sim 2900$
- Scenario 18: $X = 0.4$, $F = 0.75$, time $2900 \sim 3000$
- Scenario 19: $X = 0.4$, $F = 0.7$, time $3000 \sim 3500$
- Scenario 20: $X = 0.4$, $F = 0.65$, time $3500 \sim 3600$
- Scenario 21: $X = 0.4$, $F = 0.6$, time $3600 \sim 3700$
- Scenario 22: $X = 0.4$, $F = 0.55$, time $3700 \sim 3800$
- Scenario 23: $X = 0.4$, $F = 0.5$, time $3800 \sim 3900$
- Scenario 24: $X = 0.4$, $F = 0.45$, time $3900 \sim 4000$
- Scenario 25: $X = 0.4$, $F = 0.4$, time $4000 \sim 4500$
- Scenario 26: $X = 0.45$, $F = 0.4$, time $4500 \sim 4600$
- Scenario 27: $X = 0.5$, $F = 0.4$, time $4600 \sim 4700$
- Scenario 28: $X = 0.55$, $F = 0.4$, time $4700 \sim 4800$
- Scenario 29: $X = 0.6$, $F = 0.4$, time $4800 \sim 4900$
- Scenario 30: $X = 0.65$, $F = 0.4$, time $4900 \sim 5000$
- Scenario 31: $X = 0.7$, $F = 0.4$, time $5000 \sim 5500$
- Scenario 32: $X = 0.75$, $F = 0.4$, time $5500 \sim 5600$
- Scenario 33: $X = 0.8$, $F = 0.4$, time $5600 \sim 5700$
- Scenario 34: $X = 0.85$, $F = 0.4$, time $5700 \sim 5800$
- Scenario 35: $X = 0.9$, $F = 0.4$, time $5800 \sim 5900$
- Scenario 36: $X = 0.95$, $F = 0.4$, time $5900 \sim 6000$
- Scenario 37: $X = 1.0$, $F = 0.4$, time $6000 \sim 6500$
- Scenario 38: $X = 1.0$, $F = 0.45$, time $6500 \sim 6600$
- Scenario 39: $X = 1.0$, $F = 0.5$, time $6600 \sim 6700$
- Scenario 40: $X = 1.0$, $F = 0.55$, time $6700 \sim 6800$
- Scenario 41: $X = 1.0$, $F = 0.6$, time $6800 \sim 6900$
- Scenario 42: $X = 1.0$, $F = 0.65$, time $6900 \sim 7000$
- Scenario 43: $X = 1.0$, $F = 0.7$, time $7000 \sim 7500$
- Scenario 44: $X = 0.95$, $F = 0.7$, time $7500 \sim 7600$
- Scenario 45: $X = 0.9$, $F = 0.7$, time $7600 \sim 7700$
- Scenario 46: $X = 0.85$, $F = 0.7$, time $7700 \sim 7800$
- Scenario 47: $X = 0.8$, $F = 0.7$, time $7800 \sim 7900$
- Scenario 48: $X = 0.75$, $F = 0.7$, time $7900 \sim 8000$
- Scenario 49: $X = 0.7$, $F = 0.7$, time $8000 \sim 850$

Figure 17 shows the graphs of X, F, and MP for the ER network, WS network, and BA network. In the extended scenario, X decreases from 1 to 0.7, to 0.4, then increases to 0.7, to 1, and finally decreases to 0.7. Similarly, F first decreases from 1 to 0.7, then drops to 0.4, and finally increases back to 0.7.

First of all, from Fig. 17, when X and F are low, there are few nodes that report a match, resulting in a low match probability. Thus, the values of MP of non-adaptive algorithm on the ER network change from $MP = 1$ at time 1,

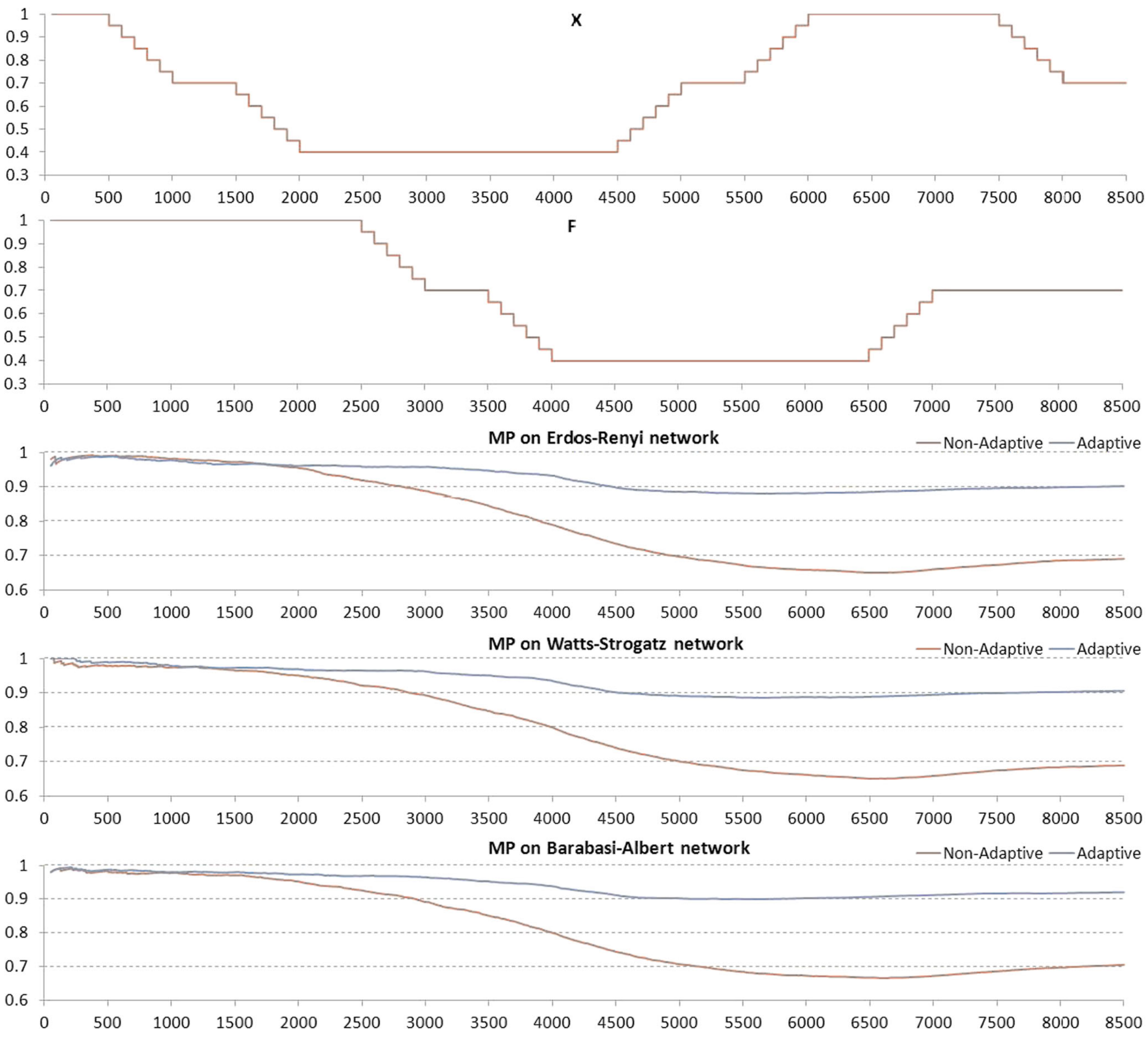

Fig. 17 Graphs of X, F, and MP of non-adaptive and adaptive algorithms, where X and F vary with $n = 10000$, $a = 14$, $l = 2$, $maxN = 1000$, $kMax = 9$, $c = 0.999$, $d = 50$, $hair = 6$, $xfGap = 0.3$, and $xfMin = 0.4$ on ER, WS, and BA networks

to $MP = 0.9820$ at time 1000, to $MP = 0.9545$ at time 2000, to $MP = 0.8877$ at time 3000, to $MP = 0.7898$ at time 4000, to $MP = 0.6969$ at time 5000, to $MP = 0.6592$ at time 6000, to $MP = 0.6602$ at time 7000, and finally to $MP = 0.6859$ at time 8000. Similarly, for the WS network, the value of MP of non-adaptive algorithm changes from $MP = 1$ at time 1, to $MP = 0.9740$ at time 1000, to $MP = 0.9505$ at time 2000, to $MP = 0.8927$ at time 3000, to $MP = 0.7998$ at time 4000, to $MP = 0.7013$ at time 5000, to $MP = 0.6627$ at time 6000, to $MP = 0.6595$ at time 7000, and finally to $MP = 0.6845$ at time 8000. Likewise, for the BA network, the value of MP of the non-adaptive algorithm changes from $MP = 1$ at time 1, to $MP = 0.9780$ at time 1000, to $MP = 0.9530$ at time 2000, to $MP = 0.8920$ at time 3000, to $MP = 0.8008$ at time 4000, to $MP = 0.7081$ at time 5000, to $MP = 0.6741$ at time 6000, to $MP = 0.6736$ at time 7000, and finally to $MP = 0.6980$ at time 8000.

Next, when the values of X and F are low, the requesting node receives fewer matches, and the consequently affects the modified chi-squared to estimate low values of X and F. Hence, my dynamic adaptive algorithm increases a and l in order to maintain high match probability MP. Subsequently, the overall value of MP of the dynamic adaptive algorithm on the ER network changes from $MP = 1$ at time 1, to $MP = 0.9780$ at time 1000, to $MP = 0.9615$ at time

Table 4 Mean value of MC and MP on ER network

	N-Adaptive	Adaptive
MP	0.6875	0.9019
MC	210	520.565

2000, to $MP = 0.9583$ at time 3000, to $MP = 0.9333$ at time 4000, to $MP = 0.8872$ at time 5000, to $MP = 0.8825$ at time 6000, to $MP = 0.8919$ at time 7000, and finally to $MP = 0.8996$ at time 8000. Similarly, the value of MP of dynamic adaptive algorithm on WS network changes from $MP = 1$ at time 1, to $MP = 0.9780$ at time 1000, to $MP = 0.9685$ at time 2000, to $MP = 0.9620$ at time 3000, to $MP = 0.9348$ at time 4000, to $MP = 0.8910$ at time 5000, to $MP = 0.8877$ at time 6000, to $MP = 0.8943$ at time 7000, and finally to $MP = 0.9023$ at time 8000. Likewise, the value of MP of dynamic adaptive algorithm on BA network changes from $MP = 1$ at time 1, to $MP = 0.9810$ at time 1000, to $MP = 0.9740$ at time 2000, to $MP = 0.9650$ at time 3000, to $MP = 0.9378$ at time 4000, to $MP = 0.9016$ at time 5000, to $MP = 0.9022$ at time 6000, to $MP = 0.9116$ at time 7000, and finally to $MP = 0.9171$ at time 8000.

From Fig. 17, I notice that the curve of MP on the ER network is quite similar to the curve of MP for both the WS and the BA network. In addition, it is clear to see that the match probability MP for the ER network, WS network, and BA network remains high even when X and F are low, and that showed the effectiveness and robustness of my dynamic adaptive algorithm.

Tables 4, 5, and 6 show the mean values of MP and MC for the entire extended scenario on the ER network, WS network, and BA network. First of all, the value of MC of the non-adaptive algorithm for all three random networks remains $MC = 210$ from the beginning to the end. Next, it is obvious to see that the mean values of MP and MC on the ER network are similar to the values of MP on both the WS and the BA network. In addition, I noticed that by applying my dynamic adaptive algorithm to the system, the mean match probability MP for each random network increases from $MP = 0.6862$ to about $MP = 0.9046$, which is quite effective. Thus, these tables confirmed the effectiveness and scalability of my dynamic adaptive algorithm, and

Table 5 Mean value of MC and MP on WS network

	N-Adaptive	Adaptive
MP	0.6862	0.9046
MC	210	544.879

Table 6 Mean value of MC and MP on BA network

	N-Adaptive	Adaptive
MP	0.7032	0.9194
MC	210	535.633

have demonstrated that a node can still find a match even when the network has low values of X and F. Furthermore, it is clear to notice that my algorithm can appropriately adjust MC to ensure high value of MP. Finally, the overall mean value of MC for all three random networks is about $MC = 544.633$, which is considered a reasonable cost.

Overall, these experiments have demonstrated the scalability and effectiveness of my algorithm in estimating the proportion of non-malicious and non-selfish nodes, and that false alarms rarely occur. When the actual proportion X of non-malicious nodes increases or decreases, my algorithm quickly estimates the changes of x of non-malicious nodes. Similarly, when the actual proportion F of non-selfish nodes increases or decreases, my algorithm quickly determines the changes of f of non-selfish nodes. In addition, I have showed that my dynamic adaptive algorithm can effectively adjust the value of a and l for achieving high match probability MP. Moreover, I have verified that when the allowable number of nodes $maxN$ in a node's view is much smaller than the actual network size n, my algorithm can still accurately estimate the value of x and f, and appropriately adjusts the number of a and l to achieve high MP. Furthermore, I considered the ER network, WS network, and BA networks in the context of my algorithm, and have showed the robustness of my algorithm on several experiments with varied network sizes. Lastly, I have constructed an extended scenario in which I varied X and F with various random networks, and have demonstrated the effectiveness and scalability of my algorithm, such that it can accurately estimate the value of x and f, appropriately adjust a and l to ensure high match probability MP, and still maintain the reasonable message cost MC.

8 Conclusion

In this paper, I have presented a dynamic adaptive message-forwarding algorithm for P2P search and retrieval system, where the main goal is to: 1) tackle the censorship and security issues; 2) employ the probability density function (pdf), Exponential Weighted Moving Average (EWMA) method, and modified chi-squared test to determine the proportion of subverted and selfish nodes; 3) defend against malicious and selective forwarding attacks by appropriately adjusting the number of requests to ensure high match probability, even when the network has large proportions of subverted and selfish nodes; and 4) guarantee robustness and scalability of my algorithm in different random networks (e.g., Erdos-Renyi, Watts-Strogatz, and Barabasi-Abert networks) and varied network sizes.

I have first explained the theory behind my dynamic adaptive message-forwarding algorithm, which uses random sampling and statistical inferences to accurately estimate the metrics that cannot be observed (e.g. malicious and selfish nodes), and thereby appropriately changes the number of requests in order to maintain a high match probability. Extensive experimental evaluations demonstrated the scalability and robustness of my algorithm, which can work well under a range of network conditions: 1) when the network has a large proportions of malicious nodes, 2) when the network has a large proportions of selfish nodes, and 3) when nodes only allow to have a mere partial view of network membership.

In the future, I plan to continue investigating and expanding my adaptive algorithms, incorporating scenarios when the network has a high rate of membership churn. In addition, I also plan to extend my work and test my algorithm when the network contains a very large membership, such as containing millions of nodes. When the network contains millions of nodes, the communication and computational costs will be larger. Therefore, I plan to investigate an algorithm that forwards messages, deals with a large number of nodes, and protects against malicious and selective forwarding attacks, but still maintains reasonable communication and computation costs.

Acknowledgments This research was supported in part by MOST 103-2410-H-194-064 and MOST 104-2410-H-194-090-MY2 of Ministry of Science and Technology, Taiwan.

References

1. Alajmi N, Elleithy K (2015) Multi-layer approach for the detection of selective forwarding attacks. Sensors 15(11):29332–29345
2. Barabási A, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512
3. Belen R (2009) Detecting disguised missing data. PhD thesis, Middle East Technical University
4. Bennett I Media censorship in china. http://www.cfr.org/china/media-censorship-china/p11515. Accessed: 2015-02-08
5. Bianchi S, Felber P, Gradinariu M (2007) Content-based Publish/subscribe using distributed r-trees. In: Proceedings of Euro-Par. Rennes, France, pp 537–548
6. Brownlee KA, Brownlee KA (1965) Statistical theory and methodology in science and engineering, vol 150. Wiley, New York
7. Busse M, Haenselmann T (2006) Wolfgang Effelsberg. Energy-efficient forwarding schemes for wireless sensor networks. In: Proceedings of the International Symposium on on World of Wireless, Mobile and Multimedia Networks, p 2006
8. Chen X, Shen J, Groves T, Wu J (2009) Probability delegation forwarding in delay tolerant networks. In: Computer Communications and Networks, 2009. ICCCN Proceedings of 18th Internatonal Conference on, p 2009
9. Chomhaill TN, McKelvey N, Curran K, Subaginy N (2015) Internet censorship in China. In: Mehdi K-P (ed) Encyclopedia of Information Science and Technology, pages pp. 1447–1451. hershey, PA: Information Science Reference third edition edition
10. Chuang YT, Melliar-Smith PM, Moser LE, Michel Lombera I (2016) Maintaining censorship resistance in the iTrust network for publication, search and retrieval. Peer-to-Peer Networking and Applications 9(2):266–283
11. Chuang YT, Lombera IM, Moser LE, Melliar-Smith PM (2011) Trustworthy distributed search and retrieval over the Internet. In: Proceedings of the International Conference on Internet Computing, pages 169–175. NV, Las Vegas
12. Clarke I, Sandberg O, Wiley B, Freenet TH (2001) An distributed anonymous information storage and retrieval system. In: Proceedings of the Workshop on Design Issues in Anonymity and Unobservability, pages 46–66, Berkeley, CA
13. Cohen B (2008) The bittorrent protocol specification
14. T. Condie, S. D. Kamvar, and H. Garcia-Molina. Adaptive peer-to-peer topologies. Inproceedings of the 4th IEEE International Conference on Peer-to-Peer Computing, pages 53–62 (August 2004) Zurich Switzerland
15. Stephen E (1990) Deering and David R Cheriton. Multicast routing in datagram internetworks and extended lans. ACM Transactions on Computer Systems (TOCS) 8(2):85–110
16. Deng H, Sun X, Wang B, Cao Y (2009) Selective forwarding attack detection using watermark in wsns. In: Computing, Communication, Control, and Management, 2009. CCCM ISECS International Colloquium on, vol 3, p 2009
17. Erd6s Paul, Rényi A (1960) On the evolution of random graphs. Publ Math Inst Hungar Acad Sci 5:17–61
18. Erramilli V, Crovella M, Chaintreau A, Diot C (2008) Delegation forwarding. In: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing, pp 251–260
19. Farley AM (1980) Broadcast time in communication networks. SIAM J Appl Math 39(2):385–390
20. Farley AM (1979) Minimal broadcast networks. Networks 9(4):313–332
21. Feller W (1968) An Introduction to Probability Theory and Its Applications, volume I John Wiley & Sons
22. Ferreira RA, Ramanathan MK, Awan A, Grama A, Jagannathan S (2005) Search with probabilistic guarantees in unstructured peer-to-peer networks. In: Proceedings of 5th IEEE International Conference on Peer-to-Peer Computing, pp 165–172. Konstanz Germany
23. Geethu PC, Mohammed AR (2013) Defense mechanism against selective forwarding attack in wireless sensor networks. In: Computing, Communications and Networking Technologies (ICCCNT) Fourth International Conference on, pages 1–4, p 2013
24. Ghosh AK, Schwartzbard A (1999) A study in using neural networks for anomaly and misuse detection. In: USENIX Security
25. Goonatilake R, Herath A, Herath S, Herath J (2007) Intrusion detection using the chi-square goodness-of-fit test for information assurance, network, forensics and software security. J Comput Sci Colleges 23(1):255–263

26. Yu G, He T (2007) Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links. In: Proceedings of the 5th international conference on Embedded networked sensor systems, pages 321–334 ACM
27. Gunturu R (2015) Survey of sybil attacks in social networks. arXiv:1504.05522
28. Gupta A, Sahin O, Agrawal D, Meghdoot A, Abbadi E (2004) Content-based publish/subscribe over P2P networks. In: Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware, pages 254–273, Toronto, Canada
29. Haas ZJ, Halpern JY, Li L (2006) Gossip-based ad hoc routing. IEEE/ACM Transac Netw (ToN) 14(3):479–491
30. Hai TranHoang, Huh E-N (2008) Detecting selective forwarding attacks in wireless sensor networks using two-hops neighbor knowledge. In: Network Computing and Applications NCA'08. Seventh, IEEE International Symposium on, pages 325–331. IEEE, p 2008
31. Hales D (2004) From selfish nodes to cooperative networks - Emergent link-based incentives in peer-to-peer networks. In: Proceedings of the Fourth International Conference on Peer-to-Peer Computing, pages 151–158, Bologna, Italy
32. Heckert A (2006) Chi-square two sample tests, September. http://www.itl.nist.gov/div898/software/dataplot/refman1/auxillar/chi2samp.htm
33. Hedetniemi SM, Hedetniemi ST, Liestman AL (1988) A survey of gossiping and broadcasting in communication networks. Networks 18(4):319–349
34. Helman P, Bhangoo J (1997) A statistically based system for prioritizing information exploration under uncertainty. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 27(4):449–466
35. Isdal T, Piatek M, Krishnamurthy A, Anderson T (2010) Privacy preserving P2P data sharing with OneSwarm. In: Proceedings of the ACM SIGCOMM Conference, pages 111–122, New Delhi, India
36. Jesi GP, Hales D, Van Steen M (2007) Identifying Malicious peers before A decentralized secure peer sampling service. In: Proceedings of the 1st International Conference on Self-Adaptive and Self-Organizing Systems, pages 237–246, Boston, MA
37. Karlof C, Wagner D (2003) Secure routing in wireless sensor networks Attacks and countermeasures. Ad Hoc Netw 1(2):293–315
38. Lazarevic A, Ertoz L, Kumar V, Ozgur A, Srivastava J (2003) A comparative study of anomaly detection schemes in network intrusion detection. In: Proceedings of the Third SIAM International Conference on Data Mining, pages 25–36, San francisco CA
39. Lee HY, Cho TH (2007) Fuzzy-based reliable data delivery for countering selective forwarding in sensor networks. In: Ubiquitous Intelligence and Computing, pages 535–544. Springer
40. Lee W, Stolfo SJ, et al. (1998) Data mining approaches for intrusion detection. In: Usenix security
41. Mathur Avijit, Newe Thomas, Rao Muzaffar (2016) Defence against black hole and selective forwarding attacks for medical wsns in the iot. Sensors 16(1):118
42. Melliar-Smith PM, Moser LE, Lombera IM, Chuang YT (2012) iTrust: Trustworthy information publication, search and retrieval. In: Proceedings of the 13th International Conference on Distributed Computing and Networking, pages 351–366, Hong Kong, China
43. Milgram S (1967) The small world problem. Psychology today 2(1):60–67
44. Mischke J, Stiller B (2004) A methodology for the design of distributed search in P2P middleware. IEEE Netw 18(1):30–37
45. Morselli R, Bhattacharjee B, Srinivasan A, Marsh MA (2005) Efficient lookup on unstructured topologies. In: Proceedings of the 24th ACM Symposium on Principles of Distributed Computing, pages 77–86, Las Vegas, NV
46. Moser LE, Melliar-Smith PM (2013) Probabilistic analysis of message forwarding. In: Proceedings of the IEEE International Conference on Computer Communications and Networks, Nassau, Bahamas
47. Moser L. E., Melliar-Smith P. M. (2014) Analysis of the match probabilities for the itrust information network with message forwarding. In: Proceedings of the International Conference on Information Networking, Phuket, Thailand, pp 340–345
48. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2007) Numerical Recipes in Fortran: The Art of Scientific Computing. Cambridge University Press. Cambridge, United Kingdom
49. Ripeanu M (2001) Peer-to-peer architecture case study: Gnutella network. In: Peer-to-Peer Computing, 2001. Proceedings. First International Conference on, pages 99–100. IEEE
50. Risson J, Moors T (2006) Survey of research towards robust peer-to-peer networks Search methods. Comput Netw: Int J Comput Telecommun Netw 50(17):3485–3521
51. Roberts SW (1959) Control chart tests based on geometric moving averages. Technometrics 1(3):239–250
52. Saeed Y, Lodhi SA, Ahmed K (2013) Obstacle management in vanet using game theory and fuzzy logic control. Int J Commun 4(1):9
53. Sasson Y, Cavin D, Schiper A (2003) Probabilistic broadcast for flooding in wireless mobile ad hoc networks conference=Wireless Communications and Networking, 2003. WCNC 2003 IEEE, volume 2, pages 1124–1130, p 2003
54. Terpstra WW, Kangasharju J, Leng C, Buchmann AP (2007) Bubblestorm: REsilient, probabilistic, and exhaustive peer-to-peer search. In: Proceedings of the ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communications, pages 49–60, Kyoto, Japan
55. Tran DA, Pham C (2010) Enabling content-based publish/subscribe services in cooperative P2P networks. Comput Netw: Int J Comput Telecommun Netw 52(11):1739–1749
56. Travers Jeffrey, Milgram Stanley (1969) An experimental study of the small world problem. Sociometry, 425–443
57. Tsoumakos D, Roussopoulos N (2003) A comparison of peer-to-peer search methods. In: Proceedings of the Sixth International Workshop on the Web and Databases, pages 61–66, San Diego CA
58. Viinikka J, Debar H (2004) Monitoring IDS background noise using EWMA control charts and alert information. In: Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection, pages 166–187, French Riviera, France
59. Wang L, Kangasharju J (2012) Real-world sybil attacks in bittorrent mainline dht. In: Global Communications Conference (GLOBECOM), 2012 IEEE, pages 826–832, IEEE
60. Watts DJ, Strogatz SH (1998) Collective dynamics of small-worldnetworks. Nature 393(6684):440–442
61. Wong B, Quasar SG (2008) A probabilistic publish-subscribe system for social networks. In: Proceedings of the 7th International Workshop on Peer-to-Peer Systems, Tampa Bay FL
62. Xiao B, Bo Y, Chemas CG (2007) Identify suspect nodes in selective forwarding attacks. J Parallel Distrib Comput 67(11):1218–1230

63. Xin-Sheng W, Yong-Zhao Z, Shu-Ming X, Liang-Min W (2009) Lightweight defense scheme against selective forwarding attacks in wireless sensor networks. In: Cyber-Enabled Distributed Computing and Knowledge Discovery CyberC'09. International Conference on, pages 226–232, IEEE, p 2009
64. Ye N, Chen Q (2001) An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. Quality and Reliability Eng Int J 17(2):105–112
65. Bo Y, Xiao B (2006) Detecting selective forwarding attacks in wireless sensor networks. In: Parallel and Distributed Processing Symposium IPDPS 2006. 20th International, pages 8–pp, IEEE, p 2006
66. Zhou B, Shi Q, Merabti M (2006) Intrusion detection in pervasive networks based on a chi-square statistic test. In: Proceedings of the 30th International Computer Software and Applications Conference, pages 203–208, Chicago, IL

Yung-Ting Chuang is an assistant professor in the Department of Information Management at National Chung Cheng University, Taiwan. Yung-Ting received all of her BS, MS, and PhD degrees in Electrical and Computer Engineering from the University of California, Santa Barbara in 2006, 2008, and 2013, respectively. Her research interests include peer-to-peer networks, social networks, distributed systems, wireless sensor networks, and information search and retrieval.