ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 3, March 2013

Security and QOS centric protocols for P2P networks: current state of the art

B.LALITHA, M.Tech
Assistant Professor
Dept. of Computer Science and Engg.
JNTUA College of Engg., Anantapur.

Abstract: Peer-to-peer (p2p) networking technologies have gained popularity as a mechanism for users to share files without the need for centralized servers. A p2p network provides a scalable and fault-tolerant mechanism to locate nodes anywhere on a network without maintaining a large amount of routing state. This allows for a variety of applications beyond simple file sharing. Examples include multicast systems, anonymous communications systems, and web caches. We envisage that an increasingly diverse set of applications will seek to make use of the growing pool of resources available around the edge of the network. here in this paper we brief the security and QOS issues of peer-to-peer networks that followed by current state of the art in handling these QOS and security issues.

Keywords: Distributed information monitoring, continual query systems, Peer-to-peer systems, P2P, structured peer-to-peer, unstructured peer-to-peer, hybrid, overlay networks, application layer service differentiation, File sharing, network coding, security, reputations, identity management.

I. Introduction:

Peer-to-peer systems, beginning with Napster, Gnutella, and several other related systems, became immensely popular in the past few years. However, under the hood, these systems represent a paradigm shift from the usual web client/server model, where there are no "servers;" every system acts as a peer, and by virtue of the huge number of peers, objects can be widely replicated, providing the opportunity for high availability and scalability, despite the lack of centralized infrastructure.

Capitalizing on this trend, researchers have defined structured peer-to-peer (p2p) overlays such as CAN [1], Chord [2], Pastry [3] and Tapestry [4] provide a self-organizing substrate for large-scale p2p applications. Unlike earlier systems, these have been subject to more extensive analysis and more careful design to guarantee scalability and efficiency. Also, rather than being designed specifically for the purpose of sharing unlawful music, these systems provide a powerful platform for the construction of a variety of decentralized services, including network storage, content distribution, web caching, searching and indexing, and application-level multicast. Structured overlays allow applications to locate any object in a probabilistically bounded, small number of network hops, while requiring per-node routing tables with only a small number of entries. As this pool of resources expands, it is likely that more diverse applications will seek to take advantage of these untapped resources, using peer-to-peer networking. We envisage that these may include ad-hoc distributed computation, streaming media on demand and multimedia group working. Each class of application raises different QoS issues. The other considerable factor is making these systems "secure", which is a significant challenge [5, 6]. In general, any system not designed to withstand an adversary is going to be broken easily by one, and p2p systems are no exception. If p2p systems are to be widely deployed on the Internet (at least, for applications beyond sharing "pirate" music files), they must be robust against a conspiracy of some nodes, acting in concert, to attack the remainder of the nodes. A malicious node might give erroneous responses to a request, both at the application level (returning false data to a query, perhaps in an attempt to censor the data) or at the network level (returning false routes, perhaps in an attempt to partition the network). Attackers might have a number of other goals, including traffic analysis against systems that try to provide anonymous communication, and censorship against systems that try to provide high availability. Unfortunately the two important factors called QOS and Security are not considered as equally important in any network models. It is obvious that if network targets QOS then it has to minimize or compromise at security provision models, vice versa need to compromise at QOS when network opting to high end of its security requirements.

The rest of the paper organized as follow:
Section II explored the classification of the security and QOS issues related to Peer-To-Peer networks. Section III revealed the solutions for these issues that are frequently quoted in literature. Current state of the art in P2P network protocols is explored in section IV, which fallowed by conclusion and references.

II. Security and QOS issues and challenges in P2P networks

P2P applications can broadly be classified [7] into the following categories based on their functionality:

- Content Distribution
- Distributed Computation
- Communication and Collaboration

Most of the real-world applications based on the P2P concept are freeware for content sharing or communication or those operating within a federated domain. Hence, QoS is not a major requirement for such applications. Hence, research on QoS for P2P networks remains in a nascent stage barring establishment of a few parameters related to content search. A new class of P2P applications integrating content, computation and communication and specialized services – the Peer Enterprises (PE) framework has been proposed by Gupta and Awasthi[8]. The PE framework has the potential to enable new cross-organizational service and business models. Hence, a detailed QoS model if available for such applications would facilitate their deployment and ensure viability. To ensure that P2P applications in general become more pervasive and can progress beyond file-sharing applications, a minimal QoS framework, based on specific quantifiable and non-quantifiable parameters must be established. Such a framework shall provide the much needed guidelines for P2P application designers and developers to cater to the QoS parameters and help meet application performance requirements effectively.

A. Quality of service issues in P2p

Concepts of P2P systems focused on QOS:

1) Content Search;
2) QoS-aware Routing;
3) Data Management
    a. Content Delivery/Video Streaming; and
    b. Use of P2P concepts in Data Grids /Service-Oriented Architectures

i. Indexing and Search

The search mechanism allows user to submit queries and receive results. Queries can be from very simple, like key lookups, to very complex, like SQL queries. The search mechanism defines the behavior of peers in terms of topology, data placement and routing. Topology defines how peers are connected. Peers may connect to everyone they wish, like Gnutella, or they may have a rigid structure. Data placement defines how data are distributed around the peers and message routing instructs how messages are propagated through the network.

Requirements of a P2P system can be specified in three main categories. The first one is expressiveness. The query language should be as detailed as possible as key lookups are not expressive enough for search over structured data. The second category is comprehensiveness. In some cases, all results are needed and not a single result. The last category is autonomy. Autonomy defines the level of restriction that is imposed to the peer. For example, in some systems, a peer can only connect with specific peers and can store its data to specific places. A final requirement is that the search mechanism is decentralized.

Apart from meeting the requirements specified above, a search mechanism should also maximize efficiency, maintain certain level of quality-of-service and assure robustness. Efficiency can be expressed in terms of resources consumed, like bandwidth, processing power or storage. Quality-of-service can de defined with multiple metrics. The most widely used metrics are number of results and response time. Robustness means the stability of the system in presence of failures.

ii. Data Management issues in P2P

Peer-to-peer (P2P) systems adopt a completely decentralized approach to resource management. By distributing data storage, processing and bandwidth across all peers in the network, they can scale without the need for powerful servers. P2P systems have been successfully used for sharing computation, e.g. SETI@home [9], communication [10] or data, e.g. Gnutella [11] and Kaaza [12]. The success of P2P systems is due to many potential benefits: scale-up to very large numbers of peers, dynamic self-organization, load balancing, parallel processing, and fault-tolerance through massive replication. Furthermore, they can be very useful in the context of mobile or pervasive computing. However, existing systems are limited to simple applications (e.g. file sharing), support limited functions (e.g. keyword search) and use simple techniques which have performance problems. Much active research is currently on-going to address the challenges posed by P2P systems in terms of highlevel data sharing services, efficiency and security. When considering data management, the main requirements of a P2P system are [13]:

• Autonomy: an autonomous peer should be able to join or leave the system at any time without restriction. It should also be able to control the data it stores and which other peers can store its data, e.g. some other trusted peers

• Query expressiveness: the query language should allow the user to describe the desired data at the appropriate level of detail. The simplest form of query is key look-up which is only appropriate for finding files. Keyword search with ranking of results is appropriate for searching documents. But for more structured data, an SQL-like query language is necessary.

• Efficiency: the efficient use of the P2P system resources (bandwidth, computing power, storage) should result in lower cost and thus higher throughput of queries, i.e. a higher number of queries can be processed by the P2P system in a given time.

• Quality of service: refers to the user-perceived efficiency of the system, e.g. completeness of query results, data consistency, data availability, query response time, etc.

• Fault-tolerance: efficiency and quality of services should be provided despite the occurrence of peers' failures. Given the dynamic nature of peers which may leave or fail at any time, the only solution is to rely on data replication.

• Security: the open nature of a P2P system makes security a major challenge since one cannot rely on trusted servers. Wrt. data management, the main security issue is access control which includes enforcing intellectual property rights on data contents.

B. Security issues in P2p

P2p systems generally require a remarkable amount of trust from their participants. A node must trust that other nodes implement the same protocols and will respect the goals of the system. In previous sections, we have discussed how mechanisms can be developed to work around a certain percent of the nodes violating the rules, but there are many other aspects where trust issues arise.

Popularity: When documents are requested based on keywords, rather than cryptographically strong hashes, it becomes possible for an adversary to spoof the results. The recording industry, in particular, has apparently been deploying "decoy" music files in p2p networks that have the same name as music files by popular artists. The decoy files have approximately the correct length, but do not contain the desired music. Similar issues have traditionally hurt search engines, where any page with a given search term inside it had an equal chance of appearing highly on the search results. The best solution to the search engine problem, as used by Google's Page-Rank technology, has been to form a notion of popularity. For Google, pages that are linked from "popular" pages are themselves more popular. An interesting issue is how to add such a notion of popularity into a p2p storage system. It might be possible to extend the audit logs, from Section 4.2, to allow nodes to indicate the value, or lack thereof, of a given file. If users can then rank each others rankings, this could potentially allow the creation of a system comparable to Google's Page-Rank.

Code: Fundamentally, p2p systems require the user to install a program on their computer that will work with other p2p nodes to implement the system. Since many applications can be built on a generic p2p substrate, an interesting issue becomes how to distribute the code to support these p2p applications. Users should not necessarily trust arbitrary programs, written by third parties, to run on their system. Recently, some commercial p2p systems were discovered to redirect sales commissions from online purchases to the p2p developers [14] and might also sell the use of CPU cycles on a user's computer to third parties, without the user getting any reimbursement [15]. Why should a user arbitrarily grant such privileges to p2p code? In many respects, this same problem occurred with active networks [16], except, in those systems, the computational model could be restricted [17]. For p2p systems, where applications can perform significant computations and consume vast amounts of disk storage, it would appear that general-purpose mobile code security architecture [18] is necessary.

III. Solutions frequently referred in literature:

P2p systems have been designed in the past to address numerous security concerns, providing anonymous communication, censorship resistance, and other features. Many such systems, including onion routing [19], Crowds [20], Publius [21], and Tangler [22], fundamentally assume a relatively small number of nodes in the network, all well-known to each other. To scale to larger numbers of nodes, where it is not possible to maintain a canonical list of the nodes in the network, additional mechanisms are necessary. Some recent p2p systems have also been developed to support censorship resistance [23] and anonymity [24, 25]. Sit and Morris [6] present a framework for performing security analyses of p2p networks. Their adversarial model allows for nodes to generate packets with arbitrary contents, but assumes that nodes cannot intercept arbitrary traffic. They then present taxonomy of possible attacks. At the routing layer, they identify node lookup, routing table maintenance and network partitioning / virtualization as security risks. They also discuss issues in higher-level protocols, such as file storage, where nodes may not necessarily maintain the necessary invariants, such as storage replication. Finally, they discuss various classes of denial-of-service attacks, including rapidly joining and leaving the network, or arranging for other nodes to send bulk volumes of data to overload a victim's network connection (i.e., distributed denial of service attacks). Dingledine et al. [26] and Douceur [27] discuss address spoofing attacks. With a large number of potentially malicious nodes in the system and without a trusted central authority to certify node identities, it becomes very difficult to know whether you can trust the claimed identity of somebody with whom you have never before communicated. Dingledine proposes to address this with various schemes, including the use of micro-cash, that allow nodes to build up reputations. Bellovin [28] identifies a number of issues with Napster and Gnutella. He discusses how difficult it might be to limit Napster and Gnutella use via firewalls, and how they can leak information those users might consider private, such as the search queries they issue to the network. Bellovin also expresses concern over Gnutella's "push" feature, intended to work around firewalls, which might be useful for distributed denial of service attacks. He considers Napster's centralized architecture to be more secure against such attacks, although it requires all users to trust the central server.

There are two established QOS centric models that are used to host peer-to-peer applications: semi-centralized and structured decentralized networks.

Semi centralized models as typified by Napster [29], SETI [9] and Kazaa [12] are based on central servers. In the case of Napster, such servers are indexing servers which store information about network nodes and the resources they are sharing. This approach is scalable, so long as there is a central authority which is capable of maintaining the necessary servers. Using this model, nodes play the client role in requesting an introduction to other nodes. From this point on, nodes are peers that are able to interact directly with one another. With semi-centralized networks, central servers are single points of failure; the failure of a single server can severely reduce the number of resources that can be discovered. Furthermore, the use of a centralized authority prevents ad-hoc resource sharing communities from being formed.

Structured and decentralized models such as Pastry [30] and Chord [31] implement a distributed hash table which provides for efficient and scalable message routing. Each network resource is assigned a unique key. Using its key, a particular resource can be contacted anywhere on the network. However, the key of a resource must be known beforehand, which makes resource discovery difficult. Attempts to build search functions onto Pastry have so far focused on the development of distributed indexing servers [32]. While this provides resource discovery, it reduces the peer-to-peer nature of the network and introduces the same legal concerns as encountered by Napster.

IV. Current State of the Art

Downloader-Initiated Random Linear Network Coding for Peer-to-Peer File Sharing : In the context of quality of service in peer-to-peer networks, Nan Wang et al[33] have investigated the effectiveness of network coding for P2P file sharing with mathematical analysis in " Downloader-Initiated Random Linear Network Coding for Peer-to-Peer File Sharing", and proposed Downloader-initiated Random Linear Network Coding to further improve the performance of network coding in the environment of P2P file-sharing networks.

As of the experiments carried out in earlier literature, the combination of p2p file sharing and random linear network coding algorithm is not well suited. This obstacle emerged in this combination is that "when network coding is carried out in the sending nodes, there is a possibility that even though a node (say, node A) has innovative information to another node (say, node B), after a random linear combination process at node A, the resultant might no longer be innovative to B". The authors referred as "unlucky combination" that initiated to

propose “Downloader-Initiated Random Network Coding” to eliminate this “unlucky combination” situation.
The rules described for “Downloader-Initiated Random Linear Network Coding” (DRLNC) algorithm as fallows.
The original data is chopped into chunks of equal size. The larger the original data, the larger the .
Each time when upload occurs, a linear combination (modulo 2) of these chunks is sent to the downloader.
Along with this linear combination, a “coefficient vector” is also sent to the downloader, informing the downloader the linear combination it is receiving.
A downloading node keeps a “coefficient matrix” of which the rows correspond to the coefficient vectors it has received.
The coefficient matrix at the source node is a identity matrix.
Each node in the network selects a limited number of other nodes as its neighbors. In this paper, this number ranges from 10 to 30.
A downloading node keeps a copy of the current coefficient matrix of each of its neighbors. A node informs the update of its coefficient matrix to its neighbors whenever it receives an innovative linear combination.
A downloading node examines its neighbors’ coefficient matrices. If the downloading node deems a neighbor node has innovative information, it then makes a request to this neighbor to download. When a downloading node makes a request to download from its neighbor node, it randomly picks a linear combination among those that carry useful information, and requests such combination explicitly from this neighbor node.
When a downloading node receives enough information such that its coefficient matrix has a rank of , it then reconstructs the original data, and changes its coefficient matrix to a k X k identity matrix like the source node.
Rule #8 above is the major differences between our proposal and other existing network coding algorithms. It eliminates the “unlucky combination” situation completely.

The simulation results explored to claim the performance comparison between proposed DRLNC and traditional coding models that uses random linear coding to upload. Here the metrics considered:
No of nodes to evaluate scalability
No of chunks in data to be transferred to evaluate stability
Number of simultaneous uploads and neighborhood size to evaluate the performance
Observation: The proposed DRLNC requires some computational overhead in the downloading nodes, such as calculating whether a neighbor node has innovative information. But the authors argued and proved in simulation results that the benefit of this extra computation can be significant to improve performance, stability and scalability. Authors in purely targeting coding aware routing, this model is not providing any solution to handle the cons when a coding aware peer is victimized by attacks. The impact of security requirements on proposed model has not been concluded. No experiments carried out to define the adaptability under high security considerations, which causes packet retransmissions, service denial for peers.

PeerCQ: Bugra Gedik et al[34] described PeerCQ in” A Scalable Peer-to-Peer Architecture for Distributed Information Monitoring Applications” , a fully decentralized peer-to-peer architecture for Internet-scale distributed information monitoring applications. The main contribution of the paper is the smart service partitioning scheme at the PeerCQ protocol layer, with the objective of achieving good load balance and good system utilization. PeerCQ poses several technical challenges in providing information monitoring services using a P2P computing paradigm. The first challenge is the need for a smart service-partitioning mechanism. The main issue regarding service partitioning is to achieve a good balance between improving the overall system utilization and maintaining the load balance among peers of the system. Several factors can affect the load balancing decision, including the computing capacity and the desired resource contribution of the peers, the willingness of peers to participate, and the characteristics of the continual queries. The second technical challenge is the reliability of CQ processing in the presence of peer departures and failures. There are three main mechanisms that make up the PeerCQ system.
The first mechanism is the overlay network membership.
The second mechanism is the PeerCQ protocol, including the service partitioning and the routing-query-based lookup algorithm.
The third mechanism is the processing of information monitoring requests in the form of continual queries (CQs).
The PeerCQ protocol specifies three important types of peer coordination:
How to find the peers that are best to serve the given information monitoring requests in terms of load balance and overall system utilization, How new nodes join the system, and How PeerCQ manages failures or departures of existing nodes.An important factor that may affect the effectiveness of the service partitioning scheme is the grouping factor. The grouping factor a is introduced at the protocol level to promote the idea of grouping similar CQs to optimize the processing of similar information monitoring requests. The grouping factor a is designed to tune the probability of assigning similar CQs to the same peer. The larger the a value is, the higher the probability that two similar CQs will be mapped to the same peer and, thus, the fewer number of CQ groups per peer. The results claimed from simulations are:
First, as the grouping factor increases, both the mean peer load and the average network cost decreases. Increasing the grouping factor helps in decreasing the mean peer load since it reduces the redundant computation by enabling more group processing. Optimized relaxed matching provides more effective reduction in the mean peer load due to its level-two grouping. Level-two grouping works better as the grouping factor a increases (i.e., the level-one grouping increases). Second, increasing the grouping factor also helps in decreasing the average network cost since the cost of fetching data items of interest from remote data sources is incurred only once per CQ group and serves for all CQs within the group. It is also clear that optimized relaxed matching provides more effective reduction in their average network cost due to its level-two grouping (which results in better

grouping) and its data source awareness, which incorporates the network cost of accessing the data items of a CQ that are being monitored into the service partitioning decision. Third, but not least, the decrease in the mean peer load and in the average network cost is desirable since it is an implication of better system utilization. However, if the grouping factor increases too much, then the goal of load balancing over the peers of the system will suffer.

Observation: PeerCQ is fully decentralized peer-to-peer architecture for Internet-scale distributed information monitoring applications. the main objective of this model is achieving good load balance and good system utilization. The authors succeed to described a better model for effective load balancing, but due to lack of considerations for the reputation of the nodes involving in that process, the client peers may be effected and possible to become as victim for unauthorized peers.

Diverse: Wu. C et al [36] open a new direction of research in "Diverse: Application-Layer Service Differentiation in Peer-to-Peer Communications" that treats different peer-to-peer sessions with different priorities, and present Diverse, a novel application layer approach to achieve service differentiation across different sessions.

Wu. C et al [36] argued that a careful observation of the end-to-end delay in the peer-to-peer session helps us to find fallowing cons in existing traditional models[44, 45, 46]

(1) The message queuing delay at each intermediate peer;

(2) the bandwidth share allocated by each peer for the session, which determines its transmission delay;

(3) the number of overlay hops traversed by data flows in the session from the source; and

(4) the delay on each overlay link between peers (determined by the sum of delays in the underlying IP-layer links).

In Diverse, to reduce the message queueing delay at each peer, Wu. C et al [36] introduced an application-layer priority scheduling algorithm referred as the overlay priority scheduling algorithm. Such scheduling of messages belonging to different sessions allows for differentiated queuing delays at the peers, with respect to sessions at different priority levels. To address the challenges of differentiating other delay parameters and throughput across different sessions, Wu C et al [36] introduced a priority-based optimal topology construction and bandwidth allocation algorithm, referred to as optimal bandwidth allocation algorithm. In this algorithm, key requirement is choosing better overlay paths for high-priority sessions, and number of overlay hops the paths traverse and the quality of overlay links. In this approach allocation of upload bandwidths occurs based on the priority levels, it further guarantee the bandwidth share for high-priority sessions. Diverse represents the design of efficient algorithms that implement these key insights.

Observation: The DIVERSE [36] model is an attempt to elevate that application layer service differentiation is an effective mechanism to compensate the lack of such support in the current IP Internet infrastructure. The authors succeed to address the challenges of providing differentiated service qualities for peer-to-peer communication sessions over the best effort Internet. Its optimal bandwidth allocation algorithm efficiently constructs optimal session topologies that maximize priority-based utilities. In addition, priority scheduling is applied at the peers in the application layer, which further improves the service quality experienced by high-priority sessions. The proposed model DIVERSE is targeting the allocation of bandwidth based on application layer level requirements, hence it is necessary to verify the scalability of the DIVERSE under nodes victimized by DOS and Replay attacks. The discussions and simulation results carried out in this paper[36] is not considering this DOS and Replay attack situations.

Random Network Coding in Peer-to-Peer Networks: Baochun Li et al [37] discussed a technique "Random Network Coding in Peer-to-Peer Networks" by aiming QOS in routing, but the impact of victimized peers that are involving in coding aware not discussed in this paper.

Baochun Li et al [37] discussed the technique called "Random Network Coding in Peer-to-Peer Networks". Conceptually, this objective is similar to the classical problem of random gossiping, which considers the problem of n people spreading a rumor initially held by only one person. The difference between P2P content distribution and the problem of random gossiping is that multiple blocks need to be distributed, rather than a single rumor. Gkantsidis et al. [47] was the first to consider random network coding as a substitute for P2P that referred as random linear network coding. In this model, however, the computational complexity of network coding escalates with an increasing number of blocks. To manage such complexity, it has been proposed [48] that blocks in a file be divided into multiple generations, and network coding is only performed within the same generation. From a more theoretical perspective, Deb et al. [49] are the first to analyze the performance of using random network coding with random gossiping. The random phone call model has also been considered, where each peer chooses a random target from the entire network to upload to, obeying the constraint that only one block can be transmitted in a round. It is assumed that each peer has one of the k blocks initially. Random network coding is originally introduced to P2P content distribution systems in order to reduce the time required to successfully distribute a file to all participating peers, yet with a simplified design of the protocol involved. As a matter of fact, by unfolding the evolution of network states over a number of rounds to a discrete-time trellis graph, it has been proved by Yeung [50] that, with the use of network coding, optimal broadcast times are achieved, regardless of the network topology and transmission schedule.

Observation: There are, however, limitations to the benefits that random network coding may bring to the design of P2P content distribution protocols. Since random network coding can only be performed on blocks within the same generation, reconciliation between a pair of neighboring peers may still be necessary across the boundary of generations. In other words, rather than collecting fine-grained blocks, now a receiver only needs to collect all the distinct coarse-grained generations that constitute the file to be distributed. It mitigates the problem of locating rare portions of a file that may be less available in the entire P2P network. In practice, such a need for reconciliation, even at the coarser granularity of generations, may negatively affect the advantage of random network coding.

Providing Witness Anonymity Under Peer-to-Peer Settings: Bo Zhu et al [38] discussed a model for security that providing witness anonymity under Peer-to-Peer Settings, though the solution is highly considerable to achieve witness anonymity and passive witness accountability but the missing factor is active witness accountability. Bo Zhu et al [38] discussed a protocol called the Secure Deep Throat (SDT) for providing witness anonymity in peer-to-peer systems has been discussed. To the best of our knowledge, SDT is the first protocol that can support both aspects of witness anonymity, i.e., identity anonymity for honest peers, and accountability for peers that attempt to misuse the anonymity feature. SDT ensures the anonymity of a peer as long as she sends out only one claim (e.g., a feedback message) per peer per malicious or selfish operation type. However, if a peer sends multiple claims against the same peer for the same reason, SDT includes a tracing mechanism to identify the peer and these claims will be ignored. Note that, witness anonymity is not perfect or unconditional anonymity. It is due to the fact that there is a fundamental tradeoff between anonymity and accountability. Hence, perfect anonymity means no accountability.

In both active and passive modes, the SDT protocol includes the following four procedures: setup, registration, claim broadcasting, and public tracing. We now present an outline of the operation of SDT in the active mode. The passive mode of operation is described in Section V-A. In that mode, the claim broadcasting and public tracing procedures operate differently. During the setup phase, the OGM generates a network-wide public/secret key pair, and publishes the public key. In addition, it publishes the method for generating the tag bases, which correspond to the messages (or the meaningful content of the claim such as "user executed a malicious behavior of type") to be anonymously authenticated in the claim broadcasting procedure, and other public information, e.g., the security parameters chosen.

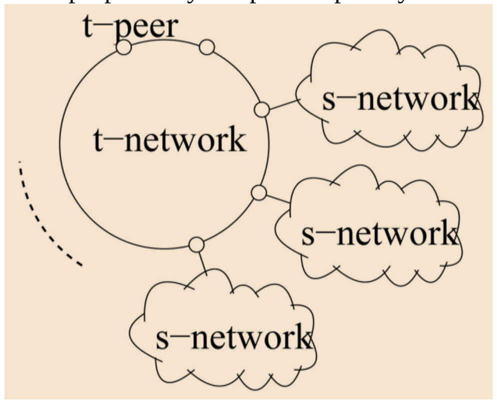
Registration is performed between the OGM and a user who wants to join the network. After this step, the user obtains a member public/secret key pair, and the OGM adds the user's identification and public key to an identification list . A user who has completed the registration procedure is called a member of the network.

Once a user detects any malicious or selfish behavior, and would like to act as a witness, i.e., broadcasting a claim bearing witness to the misbehavior, she first calculates a tag base using the method published in the setup phase. Then she generates an anonymous claim using this tag base, and broadcasts the claim through an anonymous communication system. The claim will be accepted by other users only if the sender (i.e., the witness) is a member of the network and the claim is generated using her secret key assigned in the registration procedure. Users that receive this claim store it into a local claim database, if they have not seen the same claim before. Otherwise, they simply drop the claim so as to avoid loops of forwarding. Then, the claim is forwarded using an anonymous communication system. Note that, "the same claims" are different from "the claims against the same user for the same reason." The former means those claims with exactly the same content. As to the latter, in SDT, due to a random parameter, a witness can generate different claims against the same user for the same reason, and thus these claims are considered as distinct claims.

Using LIST and the claim log, anyone can do public tracing. This procedure outputs a user ID or NO-ONE, which respectively mean "user tried to misuse witness anonymity by sending multiple claims against the same user for the same reason" and "the public tracing procedure cannot find malicious entities misusing witness anonymity."

Observation: SDT can be used in an active mode in scenarios with real-time requirements for detecting malicious behavior, e.g., for detecting and preventing the propagation of peer-to-peer worms, or it can be used in a passive mode in scenarios that do not have very strict real-time requirements, e.g., query-based reputation systems [51]. In the process of achieving witness anonymity, the proposal failed to retain active accountability [authentication of the witness], which leads to failure at prevention of witness malicious activity. But the proposed model is able to provide passive accountability [recording the witness credentials for future usage] that leads to detect the nodes those carried malicious activity.

Hybrid Peer-to-Peer System for Distributed Data Sharing: Min Yang et al [39] discussed an Efficient Hybrid Peer-to-Peer System for Distributed Data Sharing. The proposed model opted to a hybrid peer-to-peer system that composed of two parts: a core transit network and many stub networks, each of which is attached to a node in the core transit network. The core transit network, called t-network, is a structured peer-to-peer network which organizes peers into a ring similar to a chord ring. We call peers in the t-network t-peers. Each t-peer is assigned a peer ID (p id), which is a positive integer. Peers are inserted to the ring in the order of their p ids. Each t-peer maintains two pointers which point to its successor and predecessor, respectively. A finger table is also used to accelerate the search. A stub network, called s-network, is a Gnutella-style unstructured peer-to-peer network. We call the peers in an s-network s-peers except for the t-peer attached to this s-network. The topology of an s-network is arbitrarily formed. Each s-network is attached to a t-peer and this t-peer belongs to both the t-network and the s-network. One thing to mention about the s-network is that the topology of an s-network is a tree instead of a mesh. Fig. 1 shows the overview of the proposed hybrid peer-to-peer system.



ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 3, March 2013

Fig.1. Overview of the proposed hybrid peer-to-peer system[39]

Due to the imbalance of link capacity, if a peer with high link capacity is receiving messages from a peer with low link capacity, its download speed is upper bounded by the download speed of the low link capacity peer. In other words, the bandwidth of the high link capacity peer is wasted. To maximize the usage of all the link capacities, we connect the peer with higher link capacity in the system to multiple peers with lower link capacities so that the fast peer can download or upload data to the multiple slow peers simultaneously. This approach the authors referred as "Supporting Link Heterogeneity". In this proposed Hybrid peer-to-peer system, topology mismatch may increase the link stress and degrade the performance, where link stress is defined as the number of copies of a message transmitted over a certain physical link. to solve this mismatch problem proposed model adopt the idea of binning scheme introduced in [52] to construct a topology-aware overlay network.

Observation: a hybrid peer-to-peer system which combines both the structured peer-to-peer network and the unstructured peer-to-peer networks to form a two tier hierarchy to provide efficient and flexible distributed data sharing service. The top tier is the t-network which is a structured ring-based peer-to-peer network and the bottom tier is composed of multiple unstructured s-networks. As of the simulation results explored in the paper, t-network providing efficient and accurate service and s-network provide approximate best-effort service to accommodate flexibility. By assigning peers to the t-network or the s-network, the hybrid peer-to-peer system can utilize both the efficiency of the structured peer-to-peer network and the flexibility of the unstructured peer-to-peer network and achieve a good balance between them. During the node join, the computational complexity to select t-network or s-network is not concluded in simulations. The node join and node leave events would complex the key agreement protocols.

Enable Efficient Peer-to-Peer Resource Sharing in Wireless Mesh Networks: Canali.C et al [40] discussed a model to Enable Efficient Peer-to-Peer Resource Sharing in Wireless Mesh Networks. The proposed MESHCHORD assumed as (in figure 2) a two-tier architecture, the lower tier of the architecture is composed of (possibly) mobile mesh clients (clients for short), which provide (and use) the content to be shared in the P2P system; the upper tier of the architecture is composed of stationary mesh routers, which implement a DHT used to locate file/resources within the network. Unless otherwise stated, in the following we use the term peer to refer exclusively to a mesh router forming the DHT at the upper tier of the architecture. An essential assumption need to be done that is the routers are stationary and plugged, but they can be switched on/off during network lifetime. This is to account for situations that might arise in some mesh network application scenarios (e.g., community networking), in which routers might be managed by users and occasionally shut down. Also, changes in the upper tier topology might be caused by failures of some router. Mesh clients are the content users and providers: they share file/local resources with other mesh clients, as well as access resources shared by others. The DHT approach investigated in this paper is Chord [53]. Chord is based on the idea of mapping both peer (mesh router) IDs and resource IDs (keys) into the same ID space, namely, the unit ring. Each key resides on the peer with the smallest ID larger than the key. To deal with dynamic join/leaves of peers in the systems, the following procedures are implemented: When a new peer p joins the network, it first needs to initialize its predecessor and finger table. This is done by sending requests to any peer currently joining the network peer p is aware of (called hook peer). Then, the finger tables and predecessor pointers of currently active peers must be updated to account for the new peer joining the network. Finally, peer p must contact its successor s in the ring so that the key range previously managed by s can be split with p. In case no (active) hook peer can be found, the join operation fails, and the peer cannot join the Chord overlay. When an existing peer p leaves the network, it first informs its predecessor and successor in the ring about its intention of leaving the network, so that they can change their finger tables and predecessor pointers accordingly; then, peer p transfers to its successor the key range it is responsible for. In order to deal with dynamic network conditions, each active peer in the network periodically performs a Stabilize operation, which verifies and possibly updates the content of the finger table and predecessor pointer.

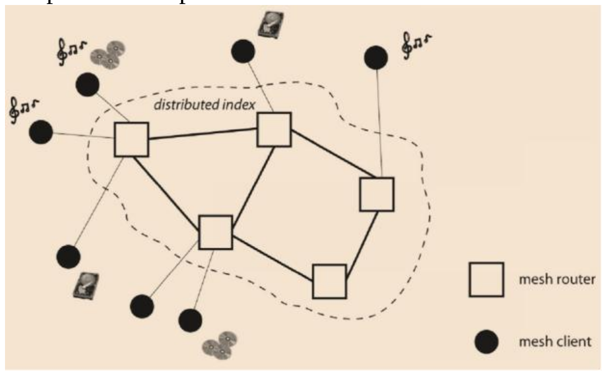


Fig 2: Two- tier architecture assumed in MESHCHORD model[40].

Observation: The performance analysis that carried out in the paper leads us to conclude that chord can be effectively used for implementing a distributed file/resource index in wireless mesh networks, at least when the coexisting traffic is low. Most importantly, Chord performance can be considerably improved by accounting for specific features of the considered application scenario, namely, precise knowledge of peer location (location awareness) and 1-hop broadcast nature of communications (MAC cross-layering). Location awareness proves very useful in reducing message overhead, of as much as 40 percent with respect to the basic Chord design. Cross-layering, on the other hand, is very effective in mitigating the negative effects of inconsistencies in the finger tables, leading to a considerable increase in the number of peers successfully joining the network. Cross layering has also a beneficial effect on the query response time, which is significantly reduced with respect to the basic Chord design. The only price to pay is a marginal increase (less than 2 percent) in the percentage of unsuccessful queries. The protocol resulting from the combination of these two techniques, MESHCHORD, sums up the relative advantages provided by location awareness and cross-layering over Chord, and considerably outperforms the basic Chord design under all respects. MESHCHORD performance gain with respect to Chord is much more evident in presence of background traffic: contrary to Chord, MESHCHORD is able to provide satisfactory performance of the P2P overlay also in presence of background traffic, while in turn increasing the performance of the applications contributing to the background traffic (e.g., increasing the average throughput of TCP flows). In general, MESHCHORD performance appears to be relatively resilient to the presence of background traffic (as long as the network is not congested), which is not the case for Chord. Although performance analysis based on simulation results has shown that MESHCHORD message overhead does not lead to network congestion by itself, overlay maintenance still requires the exchange of a relatively high number of messages in the network, which could induce performance degradation when other applications are executed concurrently with MESHCHORD. Quantifying application-layer performance degradation when several applications coexist with the P2P overlay is matter of ongoing work, as well as the problem of further reducing the message overhead induced by applications for file/ resource sharing in wireless mesh networks. The other missing issue is that performance analysis is not considering the data replication and cache management factors.

P2P Reputation Management Using Distributed Identities and Decentralized Recommendation Chains: Dewan.P et al[42] discussed "P2P Reputation Management Using Distributed Identities and Decentralized Recommendation Chains" a solution to improve the credibility of the peer that providing service. In order to participate in the reputation system, a peer needs to have a handle. The reputation of a peer is associated with its handle. This handle is commonly termed as the "identity" of the peer even though it may not "identify" a peer, i.e., it may not lead to the real-life identity of the peer. A peer receives a recommendation for each transaction performed by it, and all of its recommendations are accumulated together for calculation of the reputation of a given peer. In self-certification, The proposed system opted to a combination of self certification and identity farm approaches. Each peer's CA can generate multiple identities. The recommendations received for a peer's identity from different identities of other peers, signed by the other peer's CA(s), are identified as signed by the same CA, and are averaged to counter the liar farms. In a transaction, the requester averages all the recommendations of the provider by CAs of the provider's past recommenders. Hence, all the past recommendations owned by the provider carry equal weight but they get averaged. Finally, it adds the averages of each CA to calculate the reputation of the provider identity. Unlike the traditional CA or distributed CA-based approaches, grouping of peers preserves the anonymity of the peers; when combined with self-certification it curtails the possibility of a Sybil attack. In contrast to the traditional CA-based approach, neither the group authority nor the transacting peers can establish the identity of the peer. In addition, certificate revocations are not needed in the group-based approach as the group authority only vouches for the real-life existence of the peer, unlike the traditional certificate-based approaches where various certificate attributes are attested by the authority and necessitate revocation if any of those attributes mutate in time. If a highly reputed identity is compromised, its misuse would be self destructive as its reputation will go down if misused. Once a peer has obtained its identity, it joins the P2P network using the standard Join method of the particular P2P network. The peer (requester) searches for one or more files using the Search method provided by the network. On the basis of the responses received, as a result of its search request, the requester generates a list of peers who have the requested file(s). The number of peers who offer a particular file is denoted by RANGE. The requester selects the peer (provider) with the highest reputation from the list and initiates the cryptographic protocol. The cryptographic protocol is presented in detail in the next section. In the protocol, the requester uses the Download method of the network, to download the file from the provider. Subsequently, it verifies the integrity, authenticity, and the quality of the file. Depending on its verification results, it sends a recommendation between MIN_RECOMMENDATION and MAX_RECOMMENDATION to the provider. The recommendations are constrained to boundaries in order to make sure that one recommendation does not completely nullify or drastically improve the reputation of a provider. Once the provider receives the recommendation, it averages the previous recommendations received by it and the recent recommendation to calculate its reputation. The abovementioned steps are repeated for every transaction.

Observation: The global reputation data are protected against any malicious modification by the third party peer and are immune to any malicious modifications by their owner. The proposed protocol reduces the number of malicious transactions and consumes less bandwidth per transaction than the other reputation systems proposed in its category. It also handles the problem of highly erratic availability pattern of the peers in P2P networks. This process is not bi-directional, that is the peer that joined in network to make a request for service not been evaluated as like as a peer that providing service.

File-Sharing Preference in a Peer-to-Peer Network:
Yipeng Li et al [43] framed a model for File-Sharing Preference in a Peer-to-Peer Network. In this article investigated the fingerprint-sharing preference of users as well as the correlations between different resource categories in a real P2P file sharing system. Based on empirical data, Yipeng Li et al [43] concluded that various resource categories are strongly correlated due to users' sharing behaviors. In particular, we find that the sharing preference of users corresponds to specific resource subcategories. Empirical study conducted under the fallowing considerations.

The input dataset that considered for empirical study contains 81531 downloading logs.

A bipartite sharing graph that reflects the relations between users and resource was considered. This bipartite sharing graph highlights the interrelation among various users and resources. On one hand, connections from a certain resource to several users imply their common sharing demand.

A weighted user network was modeled by using bipartite sharing graph

A weighted resource network was modeled by using bipartite sharing graph

Observation: Based on these considerations, they analyzed hierarchical structure of weighted user network and resource network. This analysis was used to derive the user level and resource level clusters, those indicates the closeness of users in a cluster and closeness of resources in a cluster they belongs. The study finally concluded the relation between this clustering and detection user sharing preferences.

V. Conclusion

The descriptions in this paper derive the taxonomy of QOS and security issues and challenges in peer to peer network topologies. The discussion also included the current state of the art in p2p network topologies and their solutions for security and QOS challenges. As a result to this survey we can conclude as follow.

In contrast to models [33-43] evaluated in this paper, It is obvious to perform research to propose new protocols to generalize the QOS and Security requirements at their best and able to adapt the provision of QOS and security at their best.

VI. References:

[1]. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable contentaddressable network. In: Proc. ACM SIGCOMM'01, San Diego, California (2001)

[2]. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peerto- peer lookup service for Internet applications. In: Proc. ACM SIGCOMM'01, San Diego, California (2001)

[3]. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for largescale peer-to-peer systems. In: Proc. IFIP/ACM Middleware 2001, Heidelberg, Germany (2001)

[4]. Zhao, B.Y., Kubiatowicz, J.D., Joseph, A.D.: Tapestry: An infrastructure for fault-resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, U. C. Berkeley (2001)

[5]. Castro, M., Druschel, P., Ganesh, A., Rowstron, A., Wallach, D.S.: Secure routing for structured peer-to-peer overlay networks. In: Proc. OSDI 2002, Boston, Massachusetts (2002) To appear.

[6]. Sit, E., Morris, R.: Security considerations for peer-to-peer distributed hash tables. In: Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, Massachusetts (2002)

[7]. Milojicic D. J., Kalogeraki V., Lukose R., Nagaraja K., Pruyne J., Richard B., Rollins S., Xu Z., 2002, Peer-to-Peer Computing, Technical report HPL, 57, HP Labs.

[8]. Ankur Gupta, Lalit K. Awasthi: Peer Enterprises: Possibilities, Challenges and Some Ideas Towards Their Realization. OTM Workshops (2) 2007: 1011-1020

[9]. SETI@home. http://www.setiathome.ssl.berkeley.edu/.

[10]. ICQ. http://www.icq.com/.

[11]. Gnutella. http://www.gnutelliums.com/.

[12]. Kazaa. http://www.kazaa.com/.

[13]. N. Daswani, H. Garcia-Molina, B. Yang. Open problems in data-sharing peer-to-peer systems. Int. Conf. on Database Theory, 2003.

[14]. Schwartz, J., Tedeschi, B.: New software quietly diverts sales commissions. New York Times (2002) http://www.nytimes.com/2002/09/27/technology/27FREE.html.

[15]. Spring, T.: KaZaA sneakware stirs inside PCs. PC World (2002) http://www.cnn.com/2002/TECH/internet/05/07/kazaa.software.idg/index.html.

[16]. Weatherall, D.: Active network vision and reality: lessons from a capsule-based system. In: Proceedings of the Seventeenth ACM Symposium on Operating System Principles, Kiawah Island, SC (1999) 64–79

[17]. Hicks, M., Kakkar, P., Moore, J.T., Gunter, C.A., Nettles, S.: PLAN: A Packet Language for Active Networks. In: Proceedings of the Third ACM SIGPLAN International Conference on Functional Programming Languages, ACM (1998) 86–93

[18]. Wallach, D.S., Balfanz, D., Dean, D., Felten, E.W.: Extensible security architectures for Java. In: Proceedings of the Sixteenth ACM Symposium on Operating System Principles, Saint-Malo, France (1997) 116–128

[19]. Reed, M.G., Syverson, P.F., Goldschlag, D.M.: Anonymous connections and onion routing. IEEE Journal on Selected Areas in Communication: Special Issue on Copyright and Privacy Protection 16 (1998)

[20]. Reiter, M.K., Rubin, A.D.: Anonymous Web transactions with Crowds. Communications of the ACM 42 (1999) 32–48

[21]. Waldman, M., Rubin, A.D., Cranor, L.F.: Publius: A robust, tamper-evident, censorshipresistant, web publishing system. In: Proc. 9th USENIX Security Symposium, Denver, Colorado (2000) 59–72

[22]. Waldman, M., Mazires, D.: Tangler: A censorship resistant publishing system based on document entanglements. In: 8th ACM Conference on Computer and Communcation Security (CCS-8), Philadelphia, Pennsylvania (2001)

[23]. Hazel, S., Wiley, B.: Achord: A variant of the Chord lookup service for use in censorship resistant peer-to-peer. In: Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, Massachusetts (2002)

[24]. Serjantov, A.: Anonymizing censorship resistant systems. In: Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, Massachusetts (2002)

[25]. Freedman, M.J., Sit, E., Cates, J., Morris, R.: Tarzan: A peer-to-peer anonymizing network layer. In: Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, Massachusetts (2002)

[26]. Dingledine, R., Freedman, M.J., Molnar, D.: Accountability measures for peer-to-peer systems. In: Peer-to-Peer: Harnessing the Power of Disruptive Technologies, O'Reilly and Associates (2000)

[27]. Douceur, J.R.: The Sybil attack. In: Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, Massachusetts (2002)

[28]. Bellovin, S.: Security aspects of Napster and Gnutella. In: 2001 Usenix Annual Technical Conference, Boston, Massachusetts (2001) Invited talk.

[29]. Napster. www.napster.com.

[30]. Rowstron A, Druschel P. Pastry: Scalable, distributed object location and routing for largescale peer-to-peer systems, Lecture Notes in Computer Science, 2001.
[31]. Stoica I, Morris R, Karger D, Kaashoek M, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for Internet applications. Technical Report TR-819, MIT, March 2001
[32]. Reynolds P, Vahdat A. Efficient Peer-to-Peer Keyword Searching, Middleware 2003, Rio de Janeiro, Brazil. June 2003
[33]. Nan Wang; Ansari, N.; , "Downloader-Initiated Random Linear Network Coding for Peer-to-Peer File Sharing," Systems Journal, IEEE , vol.5, no.1, pp.61-69, March 2011 doi: 10.1109/JSYST.2010.2063377
[34]. Bugra Gedik and Ling Liu. 2005. A Scalable Peer-to-Peer Architecture for Distributed Information Monitoring Applications. IEEE Trans. Comput. 54, 6 (June 2005), 767-782. DOI=10.1109/TC.2005.87 http://dx.doi.org/10.1109/TC.2005.87
[35]. Jiann-Jone Chen; Chia-Jung Hu; Chun-Rong Su; , "Scalable Retrieval and Mining With Optimal Peer-to-Peer Configuration," Multimedia, IEEE Transactions on , vol.10, no.2, pp.209-220, Feb. 2008 doi: 10.1109/TMM.2007.911821
[36]. Wu, C.; Li, B.; , "Diverse: application-layer service differentiation in peer-to-peer communications," Selected Areas in Communications, IEEE Journal on , vol.25, no.1, pp.222-234, Jan. 2007 doi: 10.1109/JSAC.2007.070122
[37]. Baochun Li; Di Niu; , "Random Network Coding in Peer-to-Peer Networks: From Theory to Practice," Proceedings of the IEEE , vol.99, no.3, pp.513-523, March 2011 doi: 10.1109/JPROC.2010.2091930
[38]. Bo Zhu; Setia, S.; Jajodia, S.; Lingyu Wang; , "Providing Witness Anonymity Under Peer-to-Peer Settings," Information Forensics and Security, IEEE Transactions on , vol.5, no.2, pp.324-336, June 2010 doi: 10.1109/TIFS.2010.2041821
[39]. Min Yang; Yuanyuan Yang; , "An efficient hybrid peer-to-peer system for distributed data sharing," Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on , vol., no., pp.1-10, 14-18 April 2008 doi: 10.1109/IPDPS.2008.4536271
[40]. Canali, C.; Renda, M.E.; Santi, P.; Burresi, S.; , "Enabling Efficient Peer-to-Peer Resource Sharing in Wireless Mesh Networks," Mobile Computing, IEEE Transactions on , vol.9, no.3, pp.333-347, March 2010 doi: 10.1109/TMC.2009.134
[41]. Manghui Tu; Peng Li; I-Ling Yen; Thuraisingham, B.M.; Khan, L.; , "Secure Data Objects Replication in Data Grid," Dependable and Secure Computing, IEEE Transactions on , vol.7, no.1, pp.50-64, Jan.-March 2010 doi: 10.1109/TDSC.2008.19
[42]. Dewan, P.; Dasgupta, P.; , "P2P Reputation Management Using Distributed Identities and Decentralized Recommendation Chains," Knowledge and Data Engineering, IEEE Transactions on , vol.22, no.7, pp.1000-1013, July 2010 doi: 10.1109/TKDE.2009.45
[43]. Yipeng Li; Yong Ren; Jian Yuan; Xiuming Shan; , "File-Sharing Preference in a Peer-to-Peer Network," Circuits and Systems Magazine, IEEE , vol.11, no.1, pp.43-51, Firstquarter 2011 doi: 10.1109/MCAS.2010.939784
[44] "Bittorent," http://www.bittorrent.com.
[45] X. Zhang, J. Liu, B. Li, and T. P. Yum, "CoolStreaming/DONet: A Data-Driven Overlay Network for Live Media Streaming," in Proc. of IEEE INFOCOM 2005, March 2005.
[46] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: Peer-to-Peer Media Streaming Using CollectCast," in Proc. of ACM Multimedia 2003, November 2003.
[47] C. Gkantsidis and P. Rodriguez, BNetwork coding for large scale content distribution,[in Proc. IEEE INFOCOM, Mar. 2005, vol. 4, pp. 2235–2245.
[48] P. Chou, Y. Wu, and K. Jain, BPractical network coding,[in Proc. Allerton Conf. Commun. Control Comput., Oct. 2003. [Online]. Available: http://sites.google.com/site/saloot2/practical-network-coding.pdf
[49] S. Deb, M. Medard, and C. Choute, BAlgebraic gossip: A network coding approach to optimal multiple rumor mongering,[IEEE Trans. Inf. Theory, vol. 52, no. 6, pp. 2486–2507, Jun. 2006
[50] R. W. Yeung, BAvalanche: A network coding analysis,[Commun. Inf. Syst., vol. 7, no. 4, pp. 353–358, 2007.
[51] F. Cornelli, E. Damiani, S. De Capitani Di Vimercati, S. Paraboschi, and P. Samarati, "Choosing reputable servents in a P2P network," in Proc. 11th Int. Conf. World Wide Web, 2002, pp. 376–386
[52] S. Ratnasamy, M. Handley, R.M. Karp, and S. Shenker, "Topologically- Aware Overlay Construction and Server Selection," Proc. IEEE INFOCOM '02, June 2002.
[53] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," Proc. ACM SIGCOMM, Aug. 2001