Kademlia for data storage and retrieval in enterprise networks*

Natalya Fedotova, Stefano Fanti, Luca Veltri
Department of Information Engineering
University of Parma, Italy

Abstract — **Centralized organization of current enterprise networks doesn't represent an ideal solution in terms of information security and reliability. Denial of service, packet filtering and low resistance to failure are frequent shortcomings of centralized systems. To avoid the above problems we introduce a distributed P2P data organization system to the enterprise environment. We propose to apply Kademlia-based Distributed Hash Tables to organize data storage and retrieval systems of enterprise networks. Due to a tree-like structure of the identifier space and a prefix matching lookup algorithm, Kademlia is easy adaptable to the context of enterprise hierarchy, e.g. it facilitates the assignment of different privileges regarding data access, to various system entities. We present a solution that provides geographically separated users of enterprise networks with possibility to share, modify and publish data in parallel way.**

Keywords-component: enterprise network, Distributed Hash Tables, Kademlia, data storage and retrieval system

I. INTRODUCTION

Computer networks has become an integral part of the technical infrastructure of today enterprises, and play an important role in management of their business activity and successful realization of any type of projects.

An enterprise network is defined as a geographically dispersed network for a large business enterprise, that typically comprises a number of local area networks (LANs), which have to interface with each other, as well as with a central database management system and many client workstations. Usually, such network is based on a client-server model and centralized architecture.

An enterprise network should provide effective mechanisms for: secure communication between network terminals, rapid and secure data exchange between different system units, reliable data storage system, secure network access, simple network administration. It is also very important to make data, produced by some department maximally available for other interested entities. It should be realized in accordance with a predefined policy of attribution of different privileges, regarding data access, to users from various departments of the same organization. The possibility of collaboration between different geographically distant work groups engaged in a common task in real-time and transparent mode is indispensable for many enterprises.

* This work has been partially supported by the Italian Ministry for University and Research (MIUR) within the project PROFILES under the PRIN 2006 research program.

Currently, almost all enterprise networks as well as the groupware they utilize, have either a centralized architecture or a hybrid architecture based on the use of central servers that require high administration and maintenance costs. Moreover, centralized systems are subject to some problems regarding information security, data retrieval efficiency and reliability, such as: "denial of service" and "packet filtering" attacks, single point of failure, low network scalability.

To cope with these problems we propose to introduce a distributed peer-to-peer (P2P) data organization system to the enterprise environment. This system exploits hardware and memory resources of all terminals of the network to provide a reliable data storage system and the possibility of effective collaboration between geographically distant users. In this case all the network terminals together form a huge distributed disk space. Moreover, the distributed structure gives the possibility of incremental growth of the network, delivering complementary capacity when and where needed. This feature is very appreciable in potentially extensible environment of enterprise networks.

The solution we present here is based on resource sharing and data storage principles inhering in peer-to-peer networks based on Distributed Hash Tables (DHT). In a DHT-based P2P system a group of distributed hosts collectively manage a mapping from keys to data values according to some predefined algorithm (CAN, Chord, Pastry, Tapestry, Kademlia), without any fixed hierarchy and with a very little human assistance.

Today, a great number of P2P platforms use DHT-based overlay networks, that create a structured virtual topology above the basic transport protocol level implementing effective self-organizing data storage and lookup mechanisms. The concept of Virtual Enterprise Networks involves use of overlay mechanisms as well: Supernet layer with the appropriate address system is employed to protect data transmitted by the network layer [1].

In this paper we propose application of Kademlia DHTs to organize data storage and retrieval in enterprise networks. Due to the particular nature of the enterprise environment, it is necessary to make some modifications in Kademlia protocol in order to better suit several specific security requirements.

Since most of enterprise networks exploit trustless public network infrastructures, it is important to provide users with appropriate data authenticity and access control instruments, that can guarantee secure communication and data exchange.

We introduce a system of different levels of privileges based on use of prefix identifiers (*prefix_IDs)* for both nodes and resources to handle read/write permissions in accordance with a certain enterprise hierarchical model. This is realizable due to Kademlia's tree-like structure of the identifier space.

The rest of the paper is organized as follows: the next section provides the background on DHT principles and Kademlia protocol; it is also explained why in our case Kademlia is preferred to other DHTs. In section 3 we describe details of realization of enterprise data exchange and storage system based on Kademlia protocol. Section 4 draws some conclusions and defines directions of the future work.

II. BACKGROUND: DHT PRINCIPLES AND KADEMLIA

A. DHT routing, lookup and storage mechanisms

All DHT algorithms are based on the idea of consistent hashing and they share the following fundamental principle: route a message to a node responsible for an identifier in $O(\log_b N)$ steps using a certain routing metric, where N is the number of nodes in the system, and b is the base of the logarithm with values (2, 4, 16…).

The basic element of a typical DHT-based network is a routing table-based lookup service, which maps a given key to a node that is responsible for the key, using a hash function [2]. To publish a file, its name should be converted to a numeric key using a hash function. Then a "lookup (key)" operation should be invoked and the file with corresponding metadata should be sent to a resulting node with STORE RPC [3]. So, a node, that needs to get this file, should only convert its name into the key, invoke a "lookup (key)" and request a resulting node for a copy of the required file.

Depending on the mode of organization of the identifier space DHT-based lookup algorithms can implement routing in one dimension (Chord, Pastry, Tapestry, Kademlia) and multiple dimensions (CAN). The data structure of the routing tables maintained by existing lookup algorithms can present: skip-list (Chord), tree-like data structure (Pastry, Tapestry, Kademlia), rectangles (CAN).

B. Why Kademlia

Kademlia [3] is a peer-to-peer DHT based on the XOR metric. The distance between two identifiers is defined as: $d(x,y) = x \text{ XOR } y$. All nodes and resources in this system have 160-bit identifiers (keys). The data are replicated by finding k (the recommended value for k is 20) nodes closest to a key and storing the key/value pair on them. As it was noted above, Kademlia has a tree-like data structure and the routing process is implemented in prefix-matching mode. The routing table size is $\log_2 N$.

Comparing the main characteristics of different DHT algorithms, we can say that Kademlia is the most appropriate system to be applied to enterprise networks due to the following advantages it offers:

- the binary tree-based structure of the identifier space and routing by prefix matching permit to manage the assignment of IDs to enterprise terminals and diverse privileges to single departments in simple and intuitive mode. It also allows easy implementation of eventual algorithm modifications in the case of enhancement of a network's dimensions;
- the symmetry of XOR-metric provides peers with a possibility to learn and update routing information from queries they receive during a lookup process;
- a Kademlia system can be presented as a bucket table. So, the lookup speed can be increased by considering b bits (instead of one bit) at each step, reaching a desired resource in less time.

III. APPLICATION OF KADEMLIA PRINCIPLES IN ENTERPRISE NETWORKS ENVIRONMENT

A. Network topology

Before describing the processes of publishing and modifying data stored by network nodes, it is necessary to explain how the new identification system is organized.

Let's consider an enterprise network of some hypothetical company. We suppose that our company consists of many departments of different levels (A, B, C), which have different privileges regarding the possibility to access and modify the data produced by the same department or by another one. According to this system of privileges, any department of level A can access and modify data produced only by an office of the same level. Any *B* department has more privileges than A departments: it can get and modify data produced by any B office and also by any office of level A. Accordingly, *C* departments are enabled to access and modify files created by departments of lower levels A and B, and so on. Finally there are nodes with a manager status (M), that can get, change, store and cancel files produced by nodes of any department.

Some types of data should be accessible and shared by all the departments, for example administrative circulars, recommendations, instructions, etc. So, this information should be stored at A nodes, that belong to the lowest level of the described system, to provide free access to these resources for all the nodes of the network.

Since an enterprise network is a quite particular system with specific requirements regarding information security and access control mechanisms, it is necessary to make some appropriate modifications in Kademlia protocol to adapt this algorithm to such environment. Let's begin from keys and node IDs assignment.

B. Assignment of node identifiers

Although each workstation of an enterprise network may have a static IP address, it is not convenient to assign to a node an identifier obtained as a hash function of its static IP address, because the node would get the same node ID every time it joins the network, and it would potentially be more vulnerable to masquerade attacks.

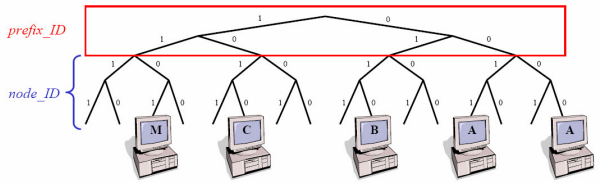

Fig. 1. Assignment of IDs to the network terminals

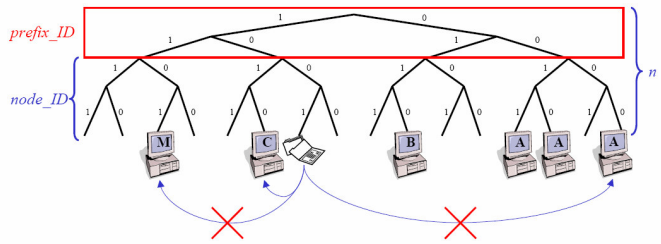

Fig 2. Example of data storage procedure

The solution we propose is a random attribution of node IDs by some trusted bootstrap terminal that should be contacted by nodes to join the network. In this case we avoid a situation when a malicious node can get and use a specific ID in order to possess certain keys related to confidential data.

To organize all work processes (storage, retrieval and exchange of data) and interactions between network terminals in accordance with hierarchical principles described above, we propose a solution explained below.

Each Kademlia node has a 160-bit node ID, that we divide in two strings of bits, the former is called prefix_ID, the latter is called node_ID. The prefix consists of β bits, and the remaining 160 - β bits represent the node_ID. The length of the prefix and of the node ID depends on the network dimensions and on the corresponding structure, i.e. on the number of nodes in the network and the number of different departments that an enterprise consists of. The prefix defines a level that a node belongs to.

When some node contacts a bootstrap terminal, this one recognizes the level of privileges the node can enjoy examining its certificate, and assigns to the node the corresponding *prefix_ID* predefined for the departments of this level. The *node_ID* should be randomly generated by the bootstrap terminal, that also verifies that the matched pair *<prefix_ID; node_ID>* doesn't coincide with some node that is already online. The assigned ID expires immediately when a node leaves the network.

Since Kademlia system has a tree-like data structure it is quite simple to realize this ID assignment technique (Fig. 1).

C. Key assignment and data storage procedures

Regarding the key assignment to files published on the network, the 160-bit key ID cannot be simply calculated as hash function of the file, as it happens in Kademlia. In fact, also in this case, the prefix-based mechanism is needed. Without involving the prefixes we can have the following situation: when a node A is going to publish some file, it first calculates the key applying a hash function to the file's content, then looks up for the nodes closest to the key. Since these ones may belong to departments of level B or C, the file could be stored at nodes of the higher level that is inaccessible for nodes of level A for the future modifications and use. So, in terms of files retrieval efficiency and according to the "competence principle" it's more useful to store data using the already introduced prefix-based approach.

In order to implement this principle, we use the same technique as for nodes' IDs, dividing a key identifier in two strings of bits that represent the prefix_ID (the first β bits) and the key_ID (the remaining 160-β bits). The key_ID can be obtained applying a hash function to the file content, as it is implemented in Kademlia. A prefix_ID of a resource usually coincides with a prefix_ID of the node that produced it. So, the prefix indicates a level of "confidentiality" of a file's content, i.e. if it is for common use or only for limited use of certain departments. However, if a node B, for example, intends to make a file it has created available for all departments, it should store the file on some node of level A. To do it the "publisher" should assign to the file a prefix_ID predefined by the bootstrap for nodes of level A. Hence, according to Kademlia principles the file will be stored at some node of level A with the node_ID "closest" to the file's key_ID. Obviously, a node is not permitted to store data at nodes of the level that is higher than its own one.

Thus, for the efficient data retrieval, files should be stored in such mode that nodes, using those files frequently during a work process, could easily get them. This is possible only if all necessary files are hosted by nodes of the same or the lower level in respect of the level of a certain node.

To illustrate the mechanisms described above we provide a simple example. In our example (Fig.2) instead of 160-bit key-space we consider 4-bit space, where the number of bits assigned to the prefix_ID is β = 2 and the other two bits represent node_ID or key_ID. All terminals represented on the figure are online.

Let's suppose that a node C needs to publish a file for internal use of department C with key_ID 01. In this case the node has to store the data at some node of level C. So the file's ID (key) will be 1001. In this example there is only one node C online apart from the publisher. Calculating the distances between the key of our resource and IDs of the active nodes according to XOR-metric, we obtain the following results:

dist(1001;1010)=0011=3

dist(1001;1101)=0100=4

dist(1001;0000)=1001=9

dist(1001;0011)=1010=10

dist(1001;0010)=1011=11

dist(1001;0110)=1111=15

So, the closest nodes are: the node C with ID=1010, the Manager node 1101, and the node 0000 of level A. But to avoid the problem of data storage at inappropriate nodes, the publisher should verify the node IDs returned by the lookup procedure and choose for the storage the nodes with opportune prefix_IDs. In our case the opportune node is C with ID=1010 of the original sub-tree. Thus, the STORE RPC should be sent only to nodes with IDs that satisfy the condition:

$$dist(ID, key) < 2^{n-\beta},$$

where n is the total number of bits in the node ID, β is the number of bits in the prefix. This inequality imposes an upper bound to the distance between the key and the target nodes. Likewise, if IDs of two nodes satisfy this condition, they reside at the same sub-tree and belong to the same department.

D. Data publication process

Publication of data in Kademlia is implemented by storing of <key, value> pairs corresponding to a certain file at nodes with the IDs closest to the key. The flexibility of DHT algorithms provides us with two possibilities: the "value" can represent information about an "address" (ID) of a node where a file can be found and the resource's description (metadata), as well as the file itself. Since, most of the files produced and exchanged by nodes in enterprise environment represent different types of documentation (text, diagrams, tables) that usually don't occupy a lot of disk space, the second way is preferred. In this way we can use mechanisms of "integrated backup" to avoid situations when some node leaves the network and resources it possesses become unreachable for other terminals. It is realized in the following mode. A publisher defines the k closest nodes for a resource to be stored according to Kademlia principles.

In order to rationally distribute network memory resources and to avoid excessive traffic increase, the "value" representing a replica of the resource is stored at γ nodes from these k nodes $(\gamma < k)$. The rest $k - \gamma$ nodes receive the STORE RPC regarding the same <key, value> pair, but with the "value" in the form of the file's metadata. So, on the network a certain number of the replicas will be presented, and leaving of some of the file's holders will not create any problem.

Besides the limit on the number of replicas γ, it also makes sense to introduce size limits for files to be replicated. The files that exceed this limit can be stored only at the nodes of origin, and the corresponding metadata should be hosted by the nodes defined by XOR metric.

To simplify the task of a publisher, we propose to apply a "tree-like data memorization model". Using this model a publisher doesn't need to send a <key, value> pair to all γ nodes. Instead of this, it sends the pair concerning a file to be stored, to two nodes that it consider the closest. In this case the <key, value> pair is complemented by a Time To Live (TTL) parameter such that:

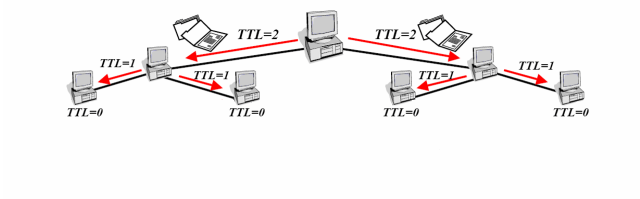

Fig.3 Tree-like data memorization model

$$\sum_{i=1}^{TTL} 2^i = \gamma$$

When the two closest nodes receive the STORE RPC, each of them store the received data and send to the node that has initialized the process a confirmation of the executed data storage. Then they verify that the TTL value is not null yet, as at each step of the process it is decremented by one. Hence, each of these nodes sends the new TTL value and the <key, value> pair to other two nodes it regards the closest to the key, and so on until TTL value has reached zero. Figure 3 illustrates a tree-like data memorization process for $\gamma=6$.

This technique of data publication represents a quite fast and reliable mechanism. Such model provides a uniform distribution of a resource's replicas within the network. It guarantees that each node potentially interested in a certain file, keeps in its k-bucket at least one contact which possesses a replica of this file or an ID of its possessor.

E. Modification and update of stored data

Now, let's consider how a node can modify some file and then publish this modification, making it available for remote users working on the same file (editing of the same document, performance of some calculations based on results of the previous step). Realization of this mechanism provides a possibility of real-time collaboration between interested users. A node that has modified some file should publish the updated version at a node from which the previous version of the file was downloaded. If the latter is not active at this moment, the file should be stored at some of the k nodes closest to the key that store the metadata or a copy of the original version of the file. In case of all these nodes leave the network, the updated file should be published in accordance with the data publication algorithm described above. As in the case of new data publication, the STORE RPCs are sent to the k closest nodes with appropriate <key, value> pairs. The tree-like data memorization model is respected in the case of updates publication too. It means that once a node has published a modified version of some file at its node of origin, this last sends the STORE RPCs with replicas and metadata of the updated file to the appropriate γ and k nodes.

To effectuate the update procedure correctly, each <key, value> pair should be supplemented with such data as:

- identity data of an "author" of modifications confirmed by his digital certificate and the node ID of the used terminal;
- exact date and time of update;
- an original file's key and ID of a node that stores it (in the case of its off-line status).

When some user publishes successive modifications of the same file, the *k* nodes that store the file or the relative metadata simply remove his precedent updates every time a new update is performed.

IV. Security mechanisms and countermeasures

A. Secure assignment of node identifiers

In section 3 we have already described the procedure of secure assignment of node identifiers that consists in random attribution of IDs by trusted bootstrap terminals. As noted above, a node with ID, that is obtained as a hash function (SHA-1) of its static IP address, is potentially vulnerable to masquerade attacks. So, we try to avoid this undesired effect applying the described ID assignment mechanism.

B. Use of certificates

For effective handling of privileges regarding data access, it is necessary that each node, before downloading or modifying some file, is able to demonstrate its belonging to a department with the privileges equal or greater than those of the file's source node. To avoid some problems regarding identity falsification, the use of digital certificates makes sense. A personal digital certificate associates a public key to some identity. Only the possessor of the certificate knows the corresponding private key, that permits him/her to create own digital signature and decrypt information encoded by the public key. In our case, such certificate is attributed to a certain network terminal and attests its identity and level of privileges it possesses. So, regarding the described enterprise model, certificates of types A, B and C enable nodes of the corresponding levels to access, store and modify data within the same-name sub-trees and those of the lower levels (for B and C nodes). The certificates of type M are assigned to the nodes with a manager status and permit them to access, store, modify and cancel data produced and stored by any other node.

C. Countermeasures

Regarding specific attacks of DHT-based environment [5], the described system proposes the following countermeasures and protective mechanisms.

Some effects of incorrect lookup rooting attacks can be avoided due to iterative character of Kademlia's lookup algorithm. In Kademlia such attacks can be detected by checking the progress of lookup at each step. In the case of absence of any progress (blatantly incorrect query forwarding), lookup process is backtracked to the previous "right" step and then proceeds with looking for an alternative direction of the search. Sybil attacks are not excluded, but lookup efficiency is improved by parallel routing (issuing α lookup requests at a time) [4].

Incorrect routing update attack can be prevented, because in Kademlia the update of routing tables is implemented by a node automatically, as a "secondary effect" of ordinary lookups and interactions with other nodes.

Partition attacks are prevented by involving trusted bootstrap terminals that should be contacted by nodes to join the network.

Mechanism of replication and storage of resources at the γ closest nodes prevents storage and retrieval attacks. So, even if one of the γ nodes maliciously denies the existence of data it is responsible for, there are other nodes enabled to provide the same resource. In this way we also eliminate a single point of failure represented by a unique node that stores a certain file. Moreover, the system can effectively cope with overload of targeted nodes with garbage packets, that represents a DHT analogue of Denial of Service attack. The tree-like data memorization model mitigates the impact of such attack due to uniform distribution of a file's replicas within the network. Since the replicas are stored in different sub-trees, even in the case of localized overload attacks to nodes of some selected part of the key-space, it is always possible to find at least one active node that stores a desired file's replica.

V. Conclusions and future work

Kademlia-based DHT represents a secure, reliable and flexible infrastructure for data storage and retrieval system in enterprise networks. We proposed some novel approaches regarding the assignment of IDs and data storage in Kademlia: prefix identifiers are introduced to handle data access privileges, and the tree-like data memorization model is proposed to improve the storage mechanisms. We are currently working toward a software implementation of the described solution.

References

[1] G. Caronni et al., "Virtual Enterprise Networks: The Next Generation of Secure Enterprise Networking", in Proceedings of the 16th Annual Computer Security Applications Conference, ACSAC 2000

[2] H. Balakrishnan et al., "Looking Up Data in P2P Systems", Communications of the ACM, Vol. 46, No. 2, pp.43-48, Feb. 2003.

[3] P. Maymounkov, D. Mazières, "Kademlia: A Peer-to-peer Information System Based on the XOR Metric", in Proceedings of the 1st International Workshop on Peer-to-peer Systems, MIT, March 2002

[4] D. Stutzbach, R. Rejaie, " Improving Lookup Performance over a Widely-Deployed DHT", in Proceedings of 25th IEEE International Conference on Computer Communications, INFOCOM'06, April 2006

[5] E. Sit, R. Morris, "Security considerations for Peer-to-Peer Distributed Hash Tables", in Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS'02), Cambridge, March 2002