On the comparison between performance of DHT-based protocols for opportunistic networks

M. C. CASTRO∗, L. GALLUCCIO†, A. KASSLER∗, S. PALAZZO†, C. RAMETTA†
∗Karlstadt University (Sweden) †DIIT - University of Catania (Italy)

Abstract: Opportunistic networks represent a new frontier for networking research as due to node mobility the network might become disconnected. Such intermittent connectivity imposes challenges to protocol design, especially when information access might require the availability of updated information about resources shared by mobile nodes. An opportunistic network can be seen as a peer-to-peer network where resources should be located in a distributed way. Numerous solutions for P2P resource management have been proposed in the last years. Among the different approaches being considered, Distributed Hash Table (DHT) based schemes offer the advantages of a distributed approach which can be tuned to network scalability. In this paper we consider two well known P2P DHT-based solutions for wireless networks denoted as Bamboo and Georoy, and compare their performance in a multihop wireless scenario. We evaluate scalability and key lookup behavior for different network sizes. The results allow us to gain insights into protocol behavior which allows to select for a given network configuration the appropriate scheme.

Keywords: Opportunistic networks, Peer-to-peer, Resource management.

1. Introduction

Opportunistic networking is a communication paradigm based on the exploitation of opportunistic inter-contact times due to users movements to disseminate resources and allow nodes communication. Opportunistic resource dissemination could be re-thought in a P2P perspective in the sense that mobile users can be identified with peers which attach and detach from the network to which they provide contents.

Reliable P2P networking represents the next frontier of wireless networks communications. In fact, on the one hand the need for employing the same services already available in a wired network, like multimedia resource sharing and, on the other hand the advocated always on connectivity and network access, make resource sharing in wireless scenarios a challenging research area especially for the limiting characteristics of the wireless medium.

The problem of efficient resource indexing in P2P networks has been studied for many years. The first ”mainstream” P2P application being released was Napster [1]. However, it cannot be considered as a ”pure” P2P system since resource indexing was managed by a central server and only data exchange by peers. In spite of its limitations, Napster paved the way for the development of many algorithms for efficient P2P resource sharing. Moreover, the spreading of wireless applications raised many new issues to be tackled in the area of wireless P2P networks, like support of node mobility and variable channel conditions, that where never faced before.

†This work was partially supported by the European Commission in the framework of the FP7 Network of Excellence in Wireless COMmunications NEWCOM++ (contract n. 216715).
One of the first pure P2P algorithms was *Gnutella v0.4* [2]. Differently from Napster, this approach does not have a central server for resource indexing but tries to find the requested resource by flooding over an unstructured overlay network which connects every node with a certain number of neighbours. Although the distributed nature of Gnutella makes it more robust than Napster, the use of flooding techniques and the presence of an unstructured overlay network limits the protocol scalability and the search efficiency. Other algorithms try to combine the efficiency of a central directory service with the robustness of a fully distributed solution in order to achieve more scalability and search accuracy, e.g. *Gnutella v0.6* and *KaZaA* [2] [3].

In those so called unstructured overlay based approaches, requests are forwarded using flooding. It has been shown that under certain circumstances, forwarding the requests to a small number of appropriate nodes instead of being flooded into the network, search efficiency and accuracy can be increased. In order to choose the "appropriate" neighbour it is necessary to give a "structure" to the overlay network. For example some algorithms use Distributed Hash Tables (DHTs) for resource indexing. In this case, each resource and each node in the network is mapped on a certain key, computed by a hash function. Every node is responsible for a range of keys and has a virtual link with a certain number of peers. When some node requests a key to a node \( n \), it compares its own ID with the key and if it falls within the node range it manages, it replies to the requester; otherwise it forwards the request to the neighbour whose ID is closer to the key. Examples of such structured overlays are *Chord* [4], *Pastry* [5] and *Tapestry* [6]. A common feature of these algorithms [4, 5, 6] is that the size of their routing tables typically increases logarithmically with the size of the network.

An interesting approach which uses an upper bounded size in the routing table is the *Viceroy* algorithm [7] which combines an unit ring topology and a butterfly [8] network topology in order to achieve a lookup performance of \( O(\log n) \) with a routing table which contains at most 7 entries.

However, none of the above solutions deals efficiently with mapping of logical and physical topology and nodes churn. These aspects will be dealt by the two protocols Bamboo and Georoy considered in this paper and discussed in the following. These protocols take inspiration from previous algorithms as Viceroy and Pastry but improve their behavior in case of network churns. The aim of this paper is to test the performance of the two protocols in a multihop scenario of varying size so as to understand when one protocol performs better than the other. Such performance comparison can be used in various ways by researchers in e.g. improving their operation or use them as a base of distributed algorithms in the opportunistic networking context.

The rest of this paper is organized as follows. In Sections 2. and 3. we give an overview of the two algorithms. In Section 4. we test the performance of the two protocols and compare their behavior. Section 5. focuses on the applicability of the two protocols to the opportunistic scenario. Finally, in Section 6. some conclusions are drawn and a discussion about future evolutions is presented.

2. Bamboo

Bamboo [9] belongs to the third generation DHT solutions. It improves previous schemes such as Pastry [5] by taking into account e.g. congestion arising due to large management traffic. While Bamboo is based on the routing logic of Pastry, management
of overlay structure is different in order to be more scalable in dynamic environments.

To maintain the network structure, Bamboo uses two set of neighbor information at each node: leafset and routing table. The leafset consists of successors and predecessors that are the numerically closest in the key space. While two nodes may be neighbors (in the leafset) in the overlay, they may be physically far away. When doing a query, the latter is forwarded until a node which has the key in its leafset to ensure correct lookup is reached. However, using only the leafset during lookups results in complexity of $O(\log N)$. To improve lookup performance, a routing table is used, which is populated with nodes that share a common prefix. Accordingly, routing table lookups are ordinary longest prefix matches.

When data is stored in the DHT using the put command, the data is routed through the DHT to the node primarily responsible for storing the data. When the responsible node gets the data, it caches it within its leafset neighbors in each direction according to the number of desired replicas. For certain applications, the number of desired replicas can cause large demands for storage space. Therefore, for data storage updates, a node periodically picks a random node in its leafset and synchronizes the stored keys with it. The correspondent node calculates the set among its stored keys that should also be stored at the sender node, sending those keys to the sender, including the hash values of the data.

The major difference between Pastry and Bamboo is the way they handle management traffic. In Pastry, management is initiated when a network change is detected, while in Bamboo management traffic is periodic regardless of network status. While reactions to changes in the routing layer operate on very small timescale, reactions to changes in overlay structure are not so fast. However, the approach to use periodic updates has shown to be beneficial during churn [9], since it does not cause management traffic bursts during congestion. Such traffic bursts can further increase packet loss probability, lead to management messages being dropped and other overlay network problems.

In order for Bamboo to be able to serve requests and maintain a consistent network view among its nodes, it needs to perform overlay maintenance message exchange between nodes. Periodic management traffic occurs at all layers of the Bamboo system. Neighbor ping is generated by every node in order to make sure that the node can still reach its one-hop neighbors in the overlay. It is also used to maintain a RTT estimate used for retransmission timeout calculations. Such timer values are used to derive, if e.g. members left the overlay or messages need to be re-transmitted. However, re-transmitting too early will lead to too high number of packets. An accurate timeout value is crucial in order to predict if a packet is lost and needs to be resent along a different path in the overlay.

Bamboo considers that two nodes share the same level when one node contains the other node in its routing table. Therefore, local routing table update is used to exchange the node information in that level. If a node gets information about other nodes that fit into the routing table, it probes the nodes to test reachability and to get a RTT estimate. If a node is reachable and fits into an empty field in the routing table, it is added. If the matching routing table entry is occupied, the node with the lowest latency is chosen.

In standard configurations, Bamboo optimizes latency. It is important to note that an optimized routing table does not influence lookup correctness, but only lookup la-
tenency [10]. As wireless networks are rather limited in bandwidth, a balance between overlay lookup efficiency and management traffic overhead is important [11]. The Bam-
boo system has been evaluated through simulation and using testbeds such as the PlanetLab [12].

3. Georoy

The Georoy algorithm [13] is a location-aware variant of the Viceroy algorithm [7]. The main target of Georoy is to build an overlay network that can provide accurate and efficient resource lookup in an ad-hoc wireless mesh network, supporting either node mobility and resources adding or removing. Using a geographic aware hash function, Georoy is able to obtain a very small stretch factor, i.e. the ratio between the hop distance of the path traversed by the query in order to find the node and the number of hops traversed in the physical network from the searching node to the searched one.

As a main difference with Chord and others, Georoy does not use a flat node topology, i.e. all nodes share resources and provide lookup facilities; instead, it uses a two level hierarchy with two different kinds of nodes:

- **Leaf Peers (LP)**: These nodes share and request resources by querying their associated super peers.

- **Super Peers (SP)**: These nodes provide the distributed resource catalog and are used by LP in order to publish and request resources.

Typically, SPs are wireless routers which are placed in the network and do not move; LPs are mobile nodes that can move and can stay connected via a handoff mechanism like in cellular networks. In Georoy, the DHT is managed only by SPs which are also responsible for the overlay construction and maintenance; so the IDs in the DHT are assigned only to these nodes. When a LP wants to share a resource it must associate this resource with a key provided by its SP according to a distributed hash function. Resource IDs are mapped in the same ID space of SPs, i.e. [0,1] so, both the SPs ID space and the resource keys space are mapped in the same interval [0,1]. Each superpeer is responsible of all IDs between its one and the one of its predecessor (which is known).

In order to provide geographic awareness, a mapping function is proposed which gives a node SP ID depending on its x and y coordinates. To explain this function we assume that nodes are deployed in a square region of side s, so all SPs are located in $R = [0, s]^2$. The mapping function $M$ is defined as follows:

$$M(x, y) = \begin{cases} \frac{s - x}{s} + \left\lfloor \frac{y}{s} \right\rfloor \frac{\Delta}{s} & \text{if } \left\lfloor \frac{y}{s} \right\rfloor \text{ is even} \\ \frac{s - x}{s} + \frac{1}{2} \left\lfloor \frac{y - \left\lfloor \frac{y}{s} \right\rfloor s}{s} \right\rfloor \frac{\Delta}{s} & \text{if } \left\lfloor \frac{y}{s} \right\rfloor \text{ is odd} \end{cases}$$

with $0 < \Delta < s$.

Using this function the $R$ space is divided in $s/\Delta$ subregions and nodes get their IDs inside these regions in increasing order, alternatively from left to right and from right to left (like a snake), in order to preserve node proximity.

When a node joins the network, it first computes its ID using the function described. Once the ID is computed, it has to choose a level $l_i$ at random between 1 and $\log n$, where $n$ is the number of nodes in the network.
Then the SP node $i$ joins the network connecting itself to the unit ring predecessor and successor and the level $l$ predecessor and successor (through a look up procedure). The unit ring predecessor and successor are the nodes with respectively the bigger ID smaller than $i$ and the smaller ID bigger than $i$, independently of the level they belong to. The level ring predecessor and successor are two nodes belonging to the same level with the same properties seen for the unit ring.

Once unit and level ring links are connected, the node must establish the butterfly connections by contacting one node at the upper level and two nodes at the lower level. The upper level link is built with the node which has the smaller ID bigger than $i$ and level $l = l_i - 1$ while the lower level links, called downward short range and downward long range links, are built with nodes that have level $l = l_i + 1$ and respectively the smaller ID bigger than $i$ and the smaller ID bigger than $i + \frac{1}{2} l_i$.

Once a SP node $i$ is connected, it can accept LPs connections and can route lookup requests. The routing follows a three step procedure: i) the request goes up in the butterfly until level 1 is reached; ii) then the request goes downward by choosing the downward link that is at the shortest distance from the requested key; iii) when there are no more downward links or both downward links overcome the requested key, the level and unit ring are chosen until the key holder is found.

The join procedure for LPs works as follows:

A LP $u$, upon entering the network, needs to invoke a join procedure to register its available catalog of resources. Accordingly, listening on the wireless interface $u$ selects the SP with the best received quality which becomes its responsible SP $p(u)$, and registers by providing it with the list of the resources it is willing to share. Such information is maintained up-to-date by $p(u)$ in a local database of available resources. We also assume that for each LP resource, having ID key $v$, there is a Home SP, $p(H)(v)$, which manages the pointer to the physical location of this latter, i.e. the current responsible superpeer the leaf peer which shares $v$ is connected to, and does not change as the LP storing $v$ moves throughout the network.

Everytime the LP moves, the responsible SP must inform the home SP of the joined LP about its new location and its resources, both available and parked. When the LP leaves the network, all resources shared are "parked" and the home SP is again informed. When the node joins again the network, the only operation required is to tag its resources as available and inform the home SP about the new location of the node.

Using both the Home SP and the responsible SP for LPs management, makes the mobility management procedure very simple and limits handoff signaling traffic. It has been proven that Georoy has a lookup complexity in the order of $O(\log n)$. For more details on Georoy procedures please refer to [13].

### 4. Performance Results

In this section, we compare performance of Bamboo with Georoy in a typical wireless multihop scenario, including the effects of routing and MAC layer on performance using ns-2.26 simulator. We deploy different scenarios by increasing the number of wireless nodes in a grid topology. The nodes are positioned at a distance of 100 meters, with 200 meters of transmission range. We use TwoRayGround propagation model and the antenna is OmniAntenna model. The transmission rate of the nodes was set to 11 Mbps, at 1 Mbps basic rate. AODV-UU routing protocol was adopted using default...
settings proposed by [14]. In our scenarios, the number of SPs varies from 25 (5x5 grid) up to 225 (15x15 grid). Simulations were performed for 500 seconds after a setup period during which nodes join the network. In this phase, nodes are building the overlay structure according to the two proposed algorithms. During the experiments, 100 random nodes are selected to acquire randomly selected keys that are located on other nodes. The lookup rate is one every five seconds. We use five repetitions and also present the standard deviation using error bars. In our tests, the following parameters were estimated: 1) Average time to perform a lookup 2) Average number of physical hops traversed by a lookup 3) Average number of logical hops traversed by a lookup 4) Ratio between physical and logical hops traversed during a lookup. The purpose of these tests is to observe the behavior of the two protocols when considering the different network scenarios. We observed that Bamboo generally performs better in terms of logical hops (see Fig. 1(a)). This is mainly because using a ring and exploiting leafset information and routing tables allows a fast ring traversal which leads to less logical hops compared to using the more complicated butterfly. An interesting observation is that the number of physical hops, which mainly determines the lookup performance in a multihop network, is in general lower when using Georoy compared to Bamboo (see Fig. 1(b)). This is because the logical and physical topologies in Georoy are tightly coupled so that the logical path is not very different from the physical. In particular, this aspect becomes more evident when the number of nodes in the network increases. This is because with an increasing number of nodes, the search complexity of the butterfly is overcome by the advantage of not crossing the entire ring during look up operations. The advantage of Georoy in reducing the number of physical hops traversed is also evident when looking at the delay figures (see Fig. 1(c)). Here, Georoy achieves better performance in terms of lookup delay especially when increasing the number of nodes.It is important to note that the total delay is composed by the key lookup time which includes the routing establishment time using AODV-UU. Georoy achieves here an overall reduction in delay of around 50% compared to Bamboo. Finally, in Fig. 1(d) we show also the routing stretch, i.e. the ratio between physical and logical hops for the two protocols. As expected, the ratio in case of Georoy is always lower than 2 while in case of Bamboo this ratio increases with increasing network size.

5. Applicability to the opportunistic scenario

This work is a first step towards a more complex comparison of the two presented DHT-based protocols, Georoy and Bamboo, to an opportunistic scenario where nodes are mobile and resource diffuse epidemically in the network. The two protocols, which employ different search procedures, can be used in wireless opportunistic scenarios where node connectivity is intermittent due to mobility of nodes and link quality variation, and resources are distributed in the network. Particularly, Georoy exploits a structured approach where only the SPs manage the resource, while the DHT algorithm and the search procedure aim at reducing the overhead due to frequent changes in the network. On the other hand, Bamboo’s overlay management is beneficial, since it avoids traffic bursts during congestion and mobility events by periodically maintaining the overlay at all layers through message exchange between nodes. The comparison of the two protocols in a multi-hop network allowed us to gain insights into protocols behavior and, in particular, on the scalability and the key lookup behavior of the two.
Figure 1: Number of hops vs. number of nodes in a) Bamboo b) Georoy c) Comparison between the lookup delay in Bamboo and Georoy d) Comparison between the two protocols in terms of ratio between Physical and Logical Hops

6. Conclusions

In this paper we have compared the performance of two DHT based P2P algorithms, namely Bamboo and Georoy. This comparison is useful to understand the general behavior of the two protocols in wireless multihop networks, including interactions with routing and MAC layer. As an overall conclusion, when the network is relatively small, Bamboo outperforms Georoy in terms of key lookup. For larger networks, Georoy shows dramatically better performance as it exploits node location when forming the overlay structure. An analysis in terms of delay and hops, both logical and physical, shows that Bamboo nicely preserves logical hops, while Georoy allows to reduce the lookup delay and keeps a mapping between logical and physical hops between 1 and 2.

Insights gained during this performance evaluation can be used as a preliminary step for setting up an opportunistic network where mobile nodes distribute contents during their movements. As an evolution of this work we are currently introducing a replication mechanism in Georoy and we plan to compare the behavior of the two schemes when also mobility and resource replication are taken into account.

References


[14] Uppsala implementation of the AODV protocol http://sourceforge.net/projects/aodvuu/files/AODV-UU/0.9.5/aodv-uu-0.9.5.tar.gz/download.