Abstract— In this paper we describe four P2P applications developed for smart phones, ranging from distributed computing to content sharing and search in closed and open user groups. Their common feature relies in the communication interface which has been cellular data transmission (SMS, HSCSD, respectively GPRS). Through our examples we highlight various aspects that application development on smart phones has to take into account, such as traffic efficiency, CPU load and social awareness. We discuss our main observations related to the experiments and highlight some promising directions for future mobile P2P applications. The discussed set of applications developed by us in the period between 2001-2005 cover the areas of distributed computing, large-scale content sharing in open communities, content sharing in closed groups, respectively keyword search in a social network of phones.

Keywords—peer-to-peer (P2P), distributed computing, content sharing, social networking

I. INTRODUCTION

Smart phones are pervasive mobile platforms to which a large number of PC applications can already be downscaled. As the computing and communication capabilities of smart phones evolve, the set of potential application types widens. The rapid evolution of smart phones was our starting point when we in late 2001 started to investigate the use of P2P applications on mobile phones. We wanted to see what kind of applications can be implemented using the existing technologies and what are the most important limitations in terms of platforms and wireless communications.

In particular, smart phones and cellular technologies are the primary focus in this paper. We were interested in comparing the performance attainable by mobile phones in the same kind of P2P applications that were widely deployed already on PC-s. As a second step, starting from the identified problems and challenges we propose ways of creating peer-to-peer applications on mobile phones taking into account the specifics of mobiles and cellular communications. Our proposals range from traffic optimization measures to creating mobile-aware network topologies and utilizing novel topologies and social awareness. Their validation has been done using simulations and proof-of-concept demonstrators.

As main challenges, in comparison to personal computers mobile phones have limited processing power, memory, communication bandwidth, battery power and graphical user interface. These limitations are also typical to a wide group of small devices.

In this paper we will summarize our findings related to four different experimental systems that probe different areas of mobile P2P. Each of the experiments highlights aspects that differentiate P2P on mobiles from PC based networks. The main focus is on the communication costs and the scarce CPU and power resources. Security and privacy considerations are also very important, but not dealt with in this contribution.

It is our belief that most of the remarks and findings can be extrapolated also to the case of P2P applications on PDA-s since the scarce communications-, CPU- and power resources are specific features of both smart phones and PDA’s.

II. EVOLUTION OF P2P FROM NETWORKS TO MOBILE DEVICES

The mainstream of peer-to-peer (P2P) applications today can be classified in three different categories: communication and collaboration, content sharing and distributed computing.

Historically, the earliest P2P protocol was probably NCP (network control protocol), the engine of the ancient Arpanet, used between mainframes. The introduction of protocols aligned to the client-server paradigm, like SMTP (simple mail transfer protocol), and later, HTTP (hypertext transfer protocol) caused a clear shift of the Internet traffic towards non-P2P traffic. But this changed radically at the beginning of the new millennium, the period in which P2P file sharing systems like Napster, Gnutella etc emerged.

Content sharing P2P systems evolve very fast: they reached today their fourth generation already. The quality of nodes has become a concern beginning from the second generation. Nodes were starting to be handled in a differentiated manner according to their “quality”. Quality of nodes is measured in several dimensions: bandwidth, online frequency, etc. This is a form of social dimension that has begun to penetrate in the area of P2P applications. We expect to see this increasingly happening when P2P gets implemented massively on mobile devices.

Aside from the mainstream P2P systems penetrated also in other application areas. P2P distributed storage networks are also a hot topic [1]. Here the exact match between the content
and the physical location is important for the quick retrieval of
the content. More recently VoIP services are implemented on
top of P2P architectures [2]. Web cache architectures are also
increasingly relying on P2P architectures [3]. And more
recently, P2P streaming radio and television also became
available [4].

In general P2P as a technique can be successful in those
cases in which

1. a feature or capacity or resource can be largely expanded
   in an economical way such as the amount of available
   resources to search and download, distributed storage or
   computational capacity,

2. there is a potential or real bottleneck to overcome, like in
   the case of heavily loaded web-servers or popular streaming
   servers,

3. there is an increasing need for communication and
   collaboration implemented in a cheap fashion or virtually free,
   like in the case of voice over IP on a P2P infrastructure and
   proximity social interactions.

Smart phones enable programmers to develop P2P clients
to almost every existing P2P network today. But smart phones
also bring new dimensions to the P2P paradigm since they are
‘always with us’. Context starts to play an important role once
the P2P node becomes mobile, resulting in new possibilities,
like exploiting our neighborhood and downloading from our
neighbors via proximity instead of downloading via expensive
computational techniques.

Proximity techniques require a different approach and will
not be covered in this contribution.

The rest of the paper is organized as follows. Section III
describes an application for distributed computing
implemented on smart phones. Section IV presents our
proposal to optimize traffic of file sharing applications taking
into account mobile-specific topologies. Section V describes a
traffic-efficient content sharing application for small, closed
groups having 100% coverage for the search. Section VI
introduces the social dimension in the overlay: a P2P keyword
search applications is outlined, built on top of the ‘phonebook
links’, creating an efficient, social-aware overlay network of
smart phones. Finally, section VI resumes our findings and
depicts possible future directions to follow for mobile P2P
applications.

III. DISTRIBUTED MOBILE P2P COMPUTING

There are certain categories of problems that can be solved
in a distributed manner. A key issue is the granularity of the
problem: if we can divide it uniformly to sub-problems in an
efficient way and assign each sub-problem to a separate node,
we can reach increased levels of supercomputing power. Good
examples for this are the SETI@home, folding@home, and the
GIMPS prime number search projects. These all fall in the
same category, relying on enthusiastic volunteers happily
offering the extra resources of their PC-s to solve problems
useful for the entire society or to some groups of researchers.

When translated to mobile peers, in addition to the costs
related to the computing platform one falls upon the
communication costs which are in general much higher than in
the case of wired networks. These costs fall in at least two
different categories: cost that we pay to the operators when we
use wireless communications and costs reflected in our
batteries running low much faster than without running the P2P
application.

We have experimented with the distributed solution of a
problem similar to the GIMPS prime number search, namely
the Mersenne prime search algorithm [5].

A Mersenne prime is a prime of the form \(2^p - 1\). The Lucas-
Lehmer test can decide if a Mersenne number is prime. The
distributed search for Mersenne numbers can be implemented
by assigning to each client an exponent \(p\) which the client
should evaluate.

We have implemented the Lucas-Lehmer test on the
Nokia9210 platform in the following fashion: a root server
distributes the task to client nodes, which can further distribute
them to their clients (see Fig.1.). This platform has limited
communication capabilities, consisting of short messaging
(SMS) and IP over HSCSD.

Computational costs are the first kind of costs that occur in
distributed computing scenarios. It is important that the task
execution time does not vary very much between tasks
assigned to different clients. Table I gives a comparison
between the used CPU resources in case of a standard PC and
the Nokia 9210i platforms in terms of needed computation time
for various Mersenne primes.

It can be seen that the PC and the communicator are very
different in terms of CPU performance. In addition, the
various exponents also mean different time execution times on
both platforms.

<table>
<thead>
<tr>
<th>Mersenne prime</th>
<th>P4/1500 MHz</th>
<th>Nokia 9210i</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2^{127} - 1)</td>
<td>16 ms</td>
<td>391 ms</td>
</tr>
<tr>
<td>(2^{4423} - 1)</td>
<td>6 s</td>
<td>5 min 43 s</td>
</tr>
<tr>
<td>(2^{9689} - 1)</td>
<td>57 s</td>
<td>52 min</td>
</tr>
<tr>
<td>(2^{19937} - 1)</td>
<td>8 min</td>
<td>6 h 53 min</td>
</tr>
</tbody>
</table>
Delays are another kind of costs that occur in a distributed computing scenario. In our test we used as communication techniques UDP (user datagram protocol) and RMI (remote method invocation) techniques, both on top of HSCSD, respectively through standard SMS.

Considering delays due to communication via cellular technologies, table II gives an overview of absolute costs (e.g. time delay) and in comparison to the CPU time needed for the computations:

From the upper part of the table results that communication costs can be considered as fixed values, therefore the mean of 0.13 s has been considered in the lower half of the table. As the exponent increases, CPU time begins to exceed largely the communications delay.

We can conclude that the algorithm granularity is not uniform in case of the Lucas-Lehmer test. Therefore this algorithm cannot be efficiently implemented on distributed networks. The task that is assigned to a node is far too heavy for the mobile phone CPU power. The computation of testing whether an exponent of relevant size results into a prime would take several years of full-time computing.

Communications generate costs not only in the form of delays, but also as fees we have to pay for the transferred bits. Communication costs of different technologies vary a lot. SMS turned out to be much more expensive than the other alternatives. For small exponents the calculation times are very short resulting into a lot of network traffic. However, when the exponent becomes bigger the cost of RMI will overtake the cost of SMS, because the calculation time per exponent is growing faster than linearly while the SMS cost is constant per exponent. An optimal, cost-efficient solution varies based on the different combinations of these aspects.

Table III presents the experienced communication costs for 3 scenarios, using RMI, SMS and UDP to evaluate exponents up to 1153.

In addition to the communication cost part of the CPU and memory of the phone has to be dedicated to the computation activity.

This decreases the performance of other applications running on the phone, which is a hidden cost. A potential solution could be intelligent multitasking, which would allow distributed computation application to be run on such low priority that it is never compromising the behavior of the other applications.

Power consumption is perhaps one of the most difficult obstacles. In general wireless communications are much more resource consuming than computations. According to [6], in TCP/IP communication over wireless 1000 to 10000 cycles of the processor per bit is needed. Running an application consumes more power than keeping the phone in a stand-by state. We experienced that it is possible to run the Mersenne application about 1 day without recharging. One possibility to handle this would be to run the application only when the battery level is high enough or when the device is connected to an external power supply.

In addition to computation, communication and battery costs we encountered in 2002 some platform-specific issues as well, most of which have been overcome today. The experiment highlighted several major requirements for the software platform. First, the software platform has to be hardware independent to make sure that the future pervasive applications can be deployed in a wide extent. Second, it should provide interoperability. Third, over the air provisioning should be supported to help the distribution of the executable tasks. Finally, it has to be thin and small enough that the API can be implemented in every small mobile device with a digital heartbeat.

As a result we believe that computationally intensive projects aiming for a general good, like new research results, are not likely to move over to mobile devices. However, applications where the task granularity is uniform enough, which would take advantage of the geographical distribution of the phones and eventually proximity techniques, and also which would provide some benefit for the user himself, are more promising directions. Examples of such applications can be mobile pathfinders using data available in mobiles scattered in a city [7] or tracking the positions of real-world objects using the emitted sound [8], in which not the computation itself is hidden in the background and the outcome is measurable through user experience.

IV. LARGE SCALE MOBILE P2P FILE SHARING

P2P content sharing is very popular. It continues generating an immense traffic on the Internet today.

P2P content sharing can be today implemented on smart phones relatively easily. For the eDonkey P2P file sharing network there is already a freeware client for Symbian smart
phones that remotely controls an eMule P2P client running on a PC [9], and more recently, a Gnutella client, called Symella [10], is also available for S60 smart phones. But these clients all plug into existing, well established communities, with the protocol, clients etc. optimized for the wired Internet, basically assuming the availability of virtually cost-free broadband data transmission.

As a direct consequence, after downloading and installing a P2P client on a mobile phone the first shock that the user experiences is the spectacular increase in the monthly fee of the GSM operator. A mobile Gnutella client running around the clock generates tremendous amount of traffic, depending of course on the community, period of the day, but very little on the user’s own traffic. The GPRS traffic indicator can be constantly around 3-4 kbps in a busy period.

On the other hand, being parts of large communities is advantageous because of the network externalities. But if a mobile P2P community could be built from scratch, then using a mobile-specific protocol and topology this could take off and be extensively used, given that smart phones already have a large penetration in the population of at least some dozens of developed countries.

In consequent we investigated a number of techniques that could enhance the communications efficiency in a cellular/wireless scenario [11]. Basis for this was Gnutella: we have taken Gnutella protocol 0.4 and modified it so that mobile-specific topologies form, instead of scale-free topologies, observed in the Internet.

The reason why scale-free topology is disadvantageous in the case of mobiles is that it requires the existence of hubs. Hubs are nodes with large node degree, handling large traffic loads. But a mobile cannot become a hub because of its inherent CPU and bandwidth limitations. Therefore other kinds of topologies should be enforced in mobile-only P2P networks. We identified 4 candidate topologies for large-scale file sharing by mobile nodes, shown in Fig. 2.

Coverage and small traffic represent a tradeoff in Gnutella kinds of networks. In our simulations we have assumed that document replication exists in order to decrease the query traffic. We have also applied an additional feature to reduce the overall traffic: we have employed adaptive TTL [11] resulting in a significant reduction of the overall traffic compared to the fix TTL rule, applied in Gnutella. TTL is a field set by the message originator and basically means the number of hops the message will travel on each path (the number of paths grows exponentially since each node forwards the message to each link).

Potential problems are the following:

(i) the person sharing the content has to pay when someone is downloading from him;

(ii) mobiles are off part of the time so more redundancy and adaptivity are needed;

(iii) users are sensitive to the processor and communication load.

And still the required bandwidth is in the order of magnitude of several kbps. So basically we are almost in the situation shown at the beginning of this section, namely as if we had plugged our mobiles to existing online communities. But additionally we need to maintain and control the network topology in order to avoid network fragmentation.

V. EFFICIENT CONTENT SHARING IN CLOSED GROUPS

First generation P2P file sharing systems all suffered from the scalability problem. Depending on their network infrastructure, one or another critical quantity turned out to grow excessively with the number of nodes in the network: server availability in Napster, traffic and message horizon in Gnutella etc.

In order to avoid bottlenecks, different techniques had been proposed. Controlled flooding and interest-based routing are some examples, which approach the problem by trying to shape the traffic, but leaving the topology unchanged.
An alternative approach relies on attempts to improve the topology. Decisions are taken at the level of a network entity: based on rules it is up to the network to decide to whom a new node might connect. The decision is thus shifted from the node to the network: this is a form of admission control, in sharp opposition with the original Gnutella concept of “preferential node might connect. The decision is thus shifted from the node topology. Decisions are taken at the level of a network entity: our earlier contributions, [13, 14].

The principle of parallel index clusters is shown in Fig. 3. Nodes are organized in index clusters. These are application layer subnets characterized by shared content metadata. Each node has a table indexing content of all the other nodes in the cluster. Therefore a search coming from outside can be responded to by one node on behalf of the entire cluster. In this way the search traffic is largely reduced. The price we have to pay is the cost of maintaining the shared list, resulting in packets whenever new content is created or old content is removed in the cluster. We call these packets index traffic in the sequel.

In addition to the mentioned search and index traffic we also have to maintain the clusters, resulting in maintenance traffic. A network based on the PIC principle can be optimized so that it minimizes the aggregate traffic, which is the sum of the three kinds of traffic. Additionally, measures can be taken such that the traffic load becomes balanced between the nodes. A careful selection of the network topology is needed for that [13].

 Networks based on PIC can vary a lot. There can be networks in which content is frequently generated but search is infrequent. A typical case is picture sharing: with the extraordinary explosion in the number of digital cameras and camera phones we are literally sunk in pictures, generating much more of them than the number of pictures that we obtain from e.g. picture search. Another extreme consists of networks in which content generation is rare, but demand for content is excessively high. It is the case of P2P music sharing and even more, the case of movie sharing and the ‘dark side’ of P2P, the so-called darknets.

A good tradeoff in a generic case, in which the weights of the different kinds of traffic in the aggregate is unknown, assuming roughly similar behavior of all nodes has been identified: the traffic per node is uniform if the nodes are distributed in clusters of equal size and if the cluster size is the square root of the number of nodes in the network. This uniform traffic is also a minimum, if we aggregate it over the entire network.

An optimum can also be found for a known ratio of traffic types: in the expression the ratio between various traffic types is also included [14].

Considering the aspect of costs, we have evaluated the amount consumed by our application if it only runs in the background. Basis for the evaluation were the rates practiced by Hungarian GSM/GPRS operators in 2004. Table V contains the outcomes of this evaluation separately for the busy hour and weekend fees. It doesn’t include the downloading costs which depend on the downloaded content.

Table V does not contain the costs of content downloading, which can be separately evaluated based on probability models of whether the found content is downloaded or not. This will depend on the size of the downloaded content that generates costs for the downloading peer and the offering peer as well.

Comparing table V with table IV it is obvious that topology control can reduce the magnitude of network traffic with two orders of magnitude in comparison to Gnutella types of search. This experiment proves that efficient content sharing is possible today in cellular networks, using techniques that reduce and possibly optimize the aggregate traffic per node. In particular, the implemented application can be utilized in sharing every kind of content available in smart phones today: text, pictures, video clips, music etc. Mobile P2P content sharing using controlled topology could really work, once the critical mass of users is reached.

### VI. SEARCH IN A SOCIAL NETWORK OF PHONES

Search for information gives not only a considerable part of the P2P traffic. The popularity of the search portals clearly indicates that keyword search is one of the most important and useful kinds of search available today.

If we were to scale down the keyword search to mobile phones, one available option is to simply use the web browser from the phone and access search portals. This way we would get the same functionality as the average PC user. However, we can enrich the functionality by using several features available in the phone.

It is obvious that the distributed application becomes richer first of all due to the social dimension available through the phone’s various features. We demonstrate this concept taking

---

<table>
<thead>
<tr>
<th>Operator fees/10 kbytes</th>
<th>0.01 EUR</th>
<th>0.035 EUR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Own query traffic/node</td>
<td>4 queries/day</td>
<td>4 queries/day</td>
</tr>
<tr>
<td>Own index update traffic/node</td>
<td>4 updates/day</td>
<td>4 updates/day</td>
</tr>
<tr>
<td>Own traffic/month</td>
<td>20 byte/s, 2.16 EUR/month</td>
<td>20 byte/s, 7.56 EUR</td>
</tr>
<tr>
<td>Cost of other user traffic/month</td>
<td>11 byte/s, 1.18 EUR/month</td>
<td>11 byte/s, 4.13 EUR/month</td>
</tr>
<tr>
<td><strong>Overall network traffic cost/node/month</strong></td>
<td><strong>3.34 EUR/month</strong></td>
<td><strong>5.685 EUR/month</strong></td>
</tr>
</tbody>
</table>
A possible use case is the following: if Alice would like to find a reliable plumber, she starts her phonebook search application and introduces in the "job" field the word “plumber”. Her phone will send short messages to a number of contacts (they will forward it to their contacts etc). On their way these messages will reach plumbers eventually. If Bob is a plumber, then his phone will recognize the match between the query and the owner’s job and reply. Finally, Alice will receive the contact details of a plumber.

We can state that the search results are “socially relevant”: the results will be persons in our social neighborhood as projected by the contact list stored in our phonebook. The better reflects the phonebook our social neighborhood, the more social relevance we get from the search.

In order to further increase the social relevance the hit quality can be added as a further dimension. A certain form of hit quality is already available in the P2P file sharing systems as well. However this does not relate to the quality of the content as source of information, but rather the quality of the connection to the provider, in terms of bandwidth and availability. But the concept can be met in more recent research work related to P2P content sharing [18].

In our approach the hit quality is represented by a ranking mechanism implemented in a context-sensitive manner. If Alice would like to know more about Bob she has the following 2 options:

1. She can find out who knows Bob,
2. She can ask a rank for Bob from these persons in a context-sensitive manner about Bob as a plumber. The concept is illustrated in Fig. 4.

It is important to remark that while the search mechanism is automatic – it does not require user intervention at all -, the ranking mechanism requires user intervention, which is appropriate from the viewpoint of privacy: there might be persons who don’t want to rank somebody or want to rank but only anonymously.

Searching a plumber in my neighborhood is just the tip of the iceberg. Such an application concept, relying on the contact list as the supporting network infrastructure, can be used to search any kind of content available in phones today, like pictures, videos and also the context history of the user, but also leads to security concerns. Widening the infrastructure to a more generic concept of the ‘contacts’ to our proximity, such as taking in our search also people we meet with frequently but we are even not aware of (but our phone is), could also widen the scope and utility of social search applications.

VII. CONCLUSIONS. THE FUTURE OF MOBILE P2P APPLICATIONS

We presented four cellular scenarios in a timely manner in which P2P applications on smart phones were implemented. Conditions for successful deployment were also assessed. Using this timeline one can attempt to identify a number of development phases through which mobile P2P applications could evolve.

It is our belief that the evolution starts with the migration of legacy P2P applications from PCs to smart phones. This is actually happening today. The essential features of the applications remain unchanged, but the “always-on, always-with-you” nature of the new platform opens new possibilities.

The evolution will continue with the exploitation of the additional features that smart phones offer. One such aspect is that smart phones will be able to take advantage of the different communication possibilities that are available in our surroundings. P2P applications in smart phones will decide which communication resource to utilize in order to optimize throughput, consumed power and communication costs. In a possible scenario, a peer could be found by following a set of phonebook links. After a while the peer acquires an IP address and the communication through the phonebook links is replaced with cellular IP communication. It allows cheaper communication and provides higher bandwidth. If the two peers later move close to each other, the cellular link is replaced with a free-of-charge proximity connection (Bluetooth, WLAN).

In a mature P2P infrastructure convergence scenario the mobiles will always use the optimal links. P2P applications could benefit through savings in bandwidth and cost, increased availability or improved user experience.

The next P2P convergence phase is a combination of proximity and social dimensions: spatial, temporal and social proximity become equivalent. These dimensions lead to numerous possibilities how a smart phone can interact both with its physical and social neighborhoods. We are surrounded with pervasive devices, communication possibilities quickly change as a function of time and location.

Ultimately we expect that smart phones and future P2P applications would be able to adapt and take advantage of pervasiveness. We would be operating in a world of multidimensional “spheres” defined around smart phones that have physical (time, space) and social (friend, colleague) dimensions. The various sensors available in smart phones extend the physical dimensions (in movement, in rain, in noisy background etc). More complex ‘sensors’ extend the social dimensions (has excited voice, is very sad, is anxious etc). The applications will eventually adapt their state machines around these dimensions to minimize cost and maximize the utility for the user.
REFERENCES

[16] B. Bakos, L. Farkas, J.K. Nurminen, Phonebook crawler – search engine for mobile P2P social networks, accepted for publication, DBA’2005, Innsbruck