PIK, Vol. 35, pp. 167–173 DOI 10.1515/pik-2012-0027

Smooth Resilient Service Provision in Large Heterogeneous Networks

Kamill Panitzek

Telekooperation/Informatik
Technische Universität Darmstadt
Hochschulstraße 10
64289 Darmstadt
Germany
panitzek@informatik.tu-darmstadt.de

Muhammad Ikram

P2P Networks Group/Informatik
Technische Universität Darmstadt
Hochschulstraße 10
64289 Darmstadt
Germany
khan@cs.tu-darmstadt.de

Max Mühlhäuser

Telekooperation/Informatik
Technische Universität Darmstadt
Hochschulstraße 10
64289 Darmstadt
Germany
max@informatik.tu-darmstadt.de

Thorsten Strufe

P2P Networks Group/Informatik
Technische Universität Darmstadt
Hochschulstraße 10
64289 Darmstadt
Germany
strufe@cs.tu-darmstadt.de

Abstract

Mobile crowd missions like multiplayer online games, disaster recovery, and mobile-assisted public gatherings gain ever more attention. They depend on highly distributed software which has to run smoothly and highly synchronous, while the underlying network – often a mix of mobile and stationary nodes – may expose Byzantine errors and malicious attacks. In our article, we present a first solution to this challenge, emphasizing first response missions as an example. It combines support for Byzantine error correction, mobile code, smooth handover, and availability and performance improvements through replication and intelligent service placement. Finally, we point at further challenges that arise on the move towards a broader spectrum of mobile crowd missions.

1 Introduction

The Internet continues to evolve rapidly; this evolution concerns all three coarse levels of consideration: the underlay network, overlay (transport, peer-to-peer, middleware etc.), and application layers.

Regarding the application layer, the most demanding kinds of applications are what we call *mobile crowd missions*: such applications interconnect a large number of users via mobile Internet access (*the mobile crowd*) – both among one another and with the stationary Internet – and require smooth continuous operation and tight synchronization. We call such applications *missions* as a discriminator from early-day, loosely coupled social networks such as electronic bulletin boards (note that modern social networks just start to expose characteristics of mobile crowd missions, tendency increasing). Examples of mobile crowd missions include mobile massive multiplayer online games, disaster recovery missions organized in real time over the network, and mobile-assisted events such as open air, street, or sports events where the mobile crowd generates and consumes private and provider-controlled media streams and acts on them in real time. We will use the example of disaster recovery as our running example in this article.

Regarding the underlay layer, mobile Internet access evolves most rapidly. It becomes faster and wider spread, and is expected to support various traffic patterns (events, data packets, streams) and communication patterns (mobile-to-base-station, mobile-to-mobile, multicast). While this evolution in the underlay network fits ideally to the evolution towards mobile crowd missions in the application layer – in fact, they are mutually dependent – the underlay continues to expose *nasty* characteristics like Byzantine errors, potentials for malicious attacks, and varying connectivity due to both fluctuating demands and varying connectivity of moving users.

This presented article addresses the following key research question: "Given the evolution as described in the underlay (fast mobile Internet access yet *nasty characteristics*) and ap-

plication layers (*mobile crowd missions*), what is the appropriate *response in between*, i.e., the appropriate support infrastructure in the overlay?"

To this end, we leverage the wealth of findings generated from research on peer-to-peer (P2P) networks, and we adopt conceptual ideas formulated by researchers working on so-called *Cloudlets* (Section 2). The support infrastructure which we propose in this article is called *P2P service overlay* which is able to disseminate services in the network thus providing a wide range of functionality to participating users. We discuss requirements, challenges, and questions that arise when designing such a system for mobile crowd missions using disaster recovery as an example (Section 3). Focusing on resilient monitoring we sketch preliminary solutions to tackle some of these challenges (Section 4). Finally, we present a short overview of our framework we developed incorporating findings presented in this article (Section 5) and end up giving concluding remarks (Section 6).

2 Prerequisites

Conventional P2P systems offer basic functionalities to locate and access data items or data streams in a decentralized fashion by connecting single network nodes (peers) with each other using an application-layer overlay. A new family of P2P systems, so-called P2P service overlays, allow for invocation, management, and migration of executable items, called mobile services. These mobile services then provide functionalities to invoking peers. Especially in mobile crowd missions such service overlays are of great benefit. For instance, P2P systems are already used to provide multiplayer online gaming service [1] and Bradler et al. evaluated different P2P overlays to use in disaster recovery on top of a mobile ad-hoc network (MANET) [2]. Important services like an alert buoy, a voice chat service, or a map service could be then provided to first responders during rescue missions using service overlays.

In our previous work, we proposed to use publicly available wireless routers (in shops, bars, etc.) to increase network resiliency of MANETs in disaster recovery [3] by creating a wireless mesh network [4] on top of these routers. One motivation for the work was the idea of so-called Cloudlets [5]. Cloudlets are nearby resource-rich computing nodes (e.g., in cafés) which could be harnessed by mobile devices over high-bandwidth, low-latency, one-hop wireless connections. P2P service overlays in combination with such an infrastructure would enable swapping of heavy tasks away from mobile nodes into more powerful Cloudlets, thus improving not only battery life but also overall system performance. Finally, computing capabilities and storage capacities of mobile devices rise each day turning smartphones into mobile Cloudlets in the future [6]. Solutions to harness all the different resources available in the future are, therefore, of great need.

To realize P2P service overlays three major prerequisites are required. These are depicted in Figure 1 and called peers, mobile services or data items, and runtime environments. *Peers* (or network nodes) are devices connected to each other, thus creating the P2P overlay. Peers are best described by their given software and hardware properties including, for example, operating system, runtime environments, CPU, memory, storage capacity, network connectivity, and also energy supply. Users interact with the entire P2P system over their respective peer including all user behavior such as real-world movement, service requests, and other interactions.

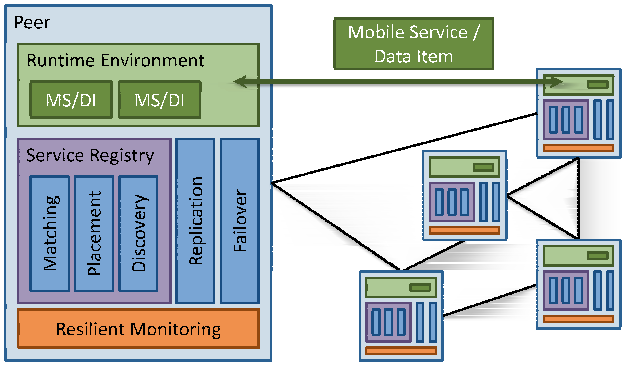

Figure 1 Main building blocks of P2P service overlays.

Mobile services or data items can be stored and distributed in the P2P system. Data items (or data streams) are already known from conventional P2P systems like files-sharing systems. Mobile services are stateful executable items providing basic functionality to users. They can be composed to more powerful applications and have the ability to be migrated between peers. P2P service overlays concentrate on the invocation and access of these stateful mobile services.

Every peer contains at least one *runtime environment* to store data items and to execute mobile services. This runtime environment can be understood as a middleware for service provision. To protect the peer from malicious code a runtime environment should be isolated from peer's hardware and software. Additionally, not to jeopardize other mobile services executed on one peer runtime environments should also be isolated from each other. For instance, a virtual machine is a possible solution to realize such an environment.

3 Building Blocks and Challenges

Looking at mobile crowd missions and disaster recovery as an example we identified basic functional and non-functional requirements of P2P service overlays. We also identified challenges associated with service overlays in such scenarios and present possible components and solutions covering them.

Service Registry: The main challenge of a P2P service overlay for mobile crowd missions is the distribution and the discovery of mobile services. The heterogeneity of devices does not allow the execution of mobile services and applications on any type of hardware and software. This is one factor that draws the invocation of mobile services problematic. Another challenge is the intelligent placement of services. Due to the mobility of participants a service must be available at changing locations. Finally, this has a big influence on how mobile services are discovered in such dynamic

environments. We call these three problems *matching*, *service placement*, and *service discovery* and since they are all related to each other we combine them to the *service registry* (cf. Figure 1).

Matching: In mobile crowd missions different device types must be handled. Due to heterogeneity of devices and provision of diverse services, peers' resources must match the requirements of mobile services. Devices cannot host services requiring other resources than are available. To provide this functionality a *matching algorithm* is needed (e.g., Xu et al. [7]). This problem can be broken down to finding the maximum flow in a bipartite graph where all mobile services and all peers are each represented by one group of nodes. A service node is connected to every peer node that is in principle able to provide the necessary resources for the service. These connections are weighted edges, where weights correspond to the costs of all needed resources (i.e., CPU, memory, etc.). A source and a sink node are inserted and connected to every service and every peer node, respectively. Finally, a maximum flow from source to sink has to be calculated so that every service is placed on one peer at most (splitting a service is not allowed). By investigating this problem we are currently working on deriving a distributed matching algorithm that is able to find suitable peers to execute available services.

Service Placement: In mobile environments services are needed at changing locations because users move while interacting with a service. Especially in MANETs the physical distance between mobile service and service consumer may introduce a large number of hops and thus a high latency among other problems. Most strategies known from literature try to reduce costs according to a fixed cost metric (usually traffic) [8–10]. For example, the REDMAN approach [9] uses heuristics to place services in the center of the network topology to minimize distances to all other nodes. Other strategies must be chosen if the positioning of services is bound to the needs of the users [11], for example, if the service is bound to a specific region (like a sensor). In 2008, Wittenburg et al. published a survey about different solutions to the service placement problem in MANETs [12]. Their conclusion was that although literature provides answers to the questions of *how many* service instances *where* to place but the question of *when* this should be done, needs to be investigated. We are working to solve this problem and are currently in the process of development and evaluations.

Service Discovery: In dynamic environments the discovery of mobile services is very challenging. Ververidis et al. published a detailed survey about service discovery in MANETs [13]. The different approaches can be classified into directory-based, directory-less, and hybrid solutions. Directory-less approaches usually use (controlled) flooding to search for services which is inefficient in most cases and also results in higher energy consumption in the case of MANETs.

In directory-based approaches a directory is used where mobile services are registered at and requested from more efficiently. Such a directory can be centralized like the UDDI [14] or a decentralized directory like a classical distributed hash table (DHT) (e.g., Chord [15] or Pastry [16]). Furthermore, there exist P2P overlays especially designed for the use in MANETs. For example, the P2P overlays Ekta [17] and MADPastry [18] build upon Pastry and MANET routing protocols, thus profiting from the benefits of a cross-layer approach. Additionally, decentralized directory-based approaches can be further optimized for performance using caching mechanisms [19], for example.

Considering hybrid approaches, Bradler et al. proposed a scalable P2P approach called PathFinder [20] combining advantages of structured and unstructured P2P systems for the first time and realizing efficient key lookup and exhaustive search. Incorporating the findings of Bradler et al. [2], we believe a super-peer overlay without indirection schemes known from typical DHTs today to be a good service discovery solution for P2P service overlays. In order to access an object using contemporary DHTs, its location must be obtained from the DHT first before accessing the object. In contrast, such a mechanism would allow for direct invocation during the service look up process if the service is not yet running or return the already invoked service instance, otherwise.

Replication and Failover: As known from today's P2P systems availability can be compromised with high churn rates (peers connecting to and disconnecting from the system). Adding peer mobility in mobile crowd missions further increases this problem due to possible connection loss resulting from limited range of wireless hardware. Therefore, a *replication mechanism* (cf. Amjad et al. [21]) is needed to automatically replicate popular services, thus increasing the availability of mobile services and also the robustness of the entire system. In P2P service overlays replicated services also include execution states which have to be synchronized continuously. If a primary service host fails a replicated service must take over, this is called *failover* [22]. Some approaches group service hosts into so-called support groups to quickly and efficiently take over if group members fail [23].

Mobile Code: Reaching high robustness to dynamic changes in the environment and to peer mobility is of great importance to P2P service overlays in general, but especially in disaster environments. Therefore, not only replication is important but also service mobility. Code and execution states must be transferred efficiently between peers. From a users' point of view this transfer has to be unnoticeable requiring smooth service provision. For instance, if users chat over a voice chat server the communication should not interrupt or stutter during while migrating the server from one peer to another. This can be achieved by using *persistent socket connections* and *code migration* (cf. Fugetta et al. [24] for an extensive overview on code mobility).

Resilient Monitoring: Finally, since resilient monitoring is a crucial component of a P2P service overlay we will describe it in more detail in the following separate section.

4 Resilient Monitoring

To improve robustness of the system, peers must be prevented from overloading, so that hosted services can be provided continuously. A *pro-active load balancing mechanism* must identify possibly overloaded peers and replicate or migrate hosted services onto other peers. More precisely, this must happen before peers overload. For example, with passive replication a primary service replicates itself onto a second

node and transmits state updates to synchronize the execution states [25]. Using information about the direct neighborhood the primary service selects a node to create the backup service on which takes over in critical situations. The assumption is that a fail is mainly due to an overload of the peer and this can be intercepted before hand. Thus, the current state of the network and the peers must be *monitored* and using this data the future peer load and network state must be *predicted*. Also, mobile services are subject to severe failures or misuses from internal or external sources. Using these monitoring and prediction components running services can be analyzed and possible countermeasures can be triggered in critical situations. Finally, a monitoring component must also oversee the functionality of the other components (cf. Figure 1) to ensure a good system performance.

In mobile crowd missions, P2P service overlays must be capable of providing persistent services to the users that are adaptive to users and devices [26], applications [27], and network-specific monitored attributes [28]. Successful deployment of P2P service overlays is based on the exploitation of the multitude of participating devices and associated data. It becomes, therefore, evident that there is a necessity to support reliable and optimal resource discovery mechanisms for P2P service overlays, which will take into account the dynamicity and heterogeneity of the underlying network infrastructure. Effective monitoring will then provide the foundation for the efficient operation of P2P service overlays.

The main requirement for any network monitoring solution regarding P2P service overlays is the support for scalability, dynamicity and heterogeneity of the network infrastructure. A prominent approach to mitigate the drawbacks brought on by such issues involves the use of P2P service overlays that constitute a networking abstraction that leads to more manageable topologies and, therefore, optimizes resource monitoring by scaling down the degree of complexity. However, the reliability of such P2P service overlays is a critical issue, in particular when the underlying network has a dynamically changing topology due to link failures and/or malicious attacks. While many solutions for designing scalable P2P service overlays have been proposed, issues such as resilience against failures and malicious attacks have not been analyzed thoroughly.

In mobile crowd missions, the monitoring component directly assess the robustness of P2P service overlay. To design resilient monitoring schemes assessing the robustness of P2P service overlay we now highlight the following key challenges:

Churn: A peer joins the P2P system when a user starts using the service offered by the P2P system. When a node joins the P2P system, it contributes some resources while making use of the resources provided by other peers and leaves the P2P system when the user stops using the service. Arrival and departure of nodes in P2P service overlays are defined as churn. Churn is one of the major factors which effect the operation of P2P service overlays. Due to high mobility and unreliable wireless connections, mobile crowd mission scenarios experience high churn and must be taken into consideration in both the design and evaluation of resilient monitoring for service overlays.

Network Failures: A peer responding to a query, is considered to be available. In case it does not respond it might not actually be unavailable, but due to common network failures it might be temporarily unreachable. It is therefore important to set an accurate timeout for a peer to answer, as well as a possible number of retries. Network connectivity failures can either be random, or systematic. In the latter case, a single peer might switch frequently from being available to being unavailable, a behavior described as flapping [29].

Edge Effects: Because observations can only be made for a finite length of time, edge effects can distort the measurements. A monitor is oblivious of any property that spans more than the observed time window, e.g., long session lengths. This adds a bias toward short sessions. To cope with this problem, Saroiu et al. [30] have introduced the create-based-method in which the measurement window is divided, and only sessions that start in the first half are considered. Using this method, an unbiased measurement of session lengths of up to half of the time frame is possible, while also identifying sessions exceeding this length. Although the mean session length is still unknown, their median can be calculated with this method.

Malicious Attacks: In mobile crowd missions, the transmission in the open medium and the reliance on untrusted nodes for P2P service overlay make it extremely vulnerable to Byzantine failures and malicious attacks [31]. For example, an attacker can forge, modify, drop, or replay routing packets, which can lead to discovering non-optimal or adversarial-controlled routes [31]. In addition to attacks against routing, monitoring schemes in P2P service overlays are also vulnerable to attacks. For instance, a malicious client can perform arbitrary actions, the consequences are localized to operations on their own node that affect other components of the P2P service overlay. An attacker can create much more damage to the monitoring scheme and hence to the P2P service overlay by not adhering to the monitoring protocol. In order to ensure the reliability of P2P service overlays monitoring must be resilient to circumvent malicious attacks in the system.

In literature, meshes [32, 33] and trees [34, 35] constitute the two prominent monitoring topologies for a decentralized environment. In a mesh-based monitoring the monitored information, also called *attributes*, are exchanged among participating peers [36]. The attributes exchange is normally held randomly among peers. One or several neighbors are randomly chosen to exchange monitored information. In some cases, gossip-based communication is used which is often used as a synonym when describing the communication within mesh-based monitoring mechanisms. The communication links are limited to the nearest neighborhood, i.e. physical or overlay neighborhood. Both mesh- and gossip-based monitoring schemes are suitable in decentralized environments with no well-defined structure or dynamic structure. In tree-based monitoring schemes, the tree is build on top of well-defined or stable structure of the communication network. In such schemes, the information is only exchanged between peers who are direct children and parents. Furthermore, there are several hybrid approaches, combining gossip-based aggregation in trees [28] or creating trees of mesh-based networks [37, 38].

Topology maintenance of P2P service overlays depend on the underlying network environment and its network topology. Monitoring approaches that are deployed in static and structured environments, such as in the area of grid computing [38], heavily differ from approaches for autonomous systems where users have high mobility, peer dynamics, and are subject to Byzantine attacks [31]. P2P service overlays must actively maintain the monitoring topology and manage high churn [37, 39], Byzantine errors, and potential malicious attacks [31].

5 First Steps Towards a Service Overlay

As a feasibility study, we already implemented a Java-based framework called *PeerMoS* providing some of the features and components described above. We used OpenChord [40] as service registry based on the Chord DHT by Stoica et al. [15]. Service placement and service discovery are both also realized using the Chord DHT where service locations are stored in the Chord DHT and can be accessed upon service requests. In our first version of PeerMoS, we assumed homogeneous peers and did not implement a matching component. Since PeerMoS builds upon mobile services we realized code migration using software agents from the MundoCore framework [41]. Services are then implemented using an agent as a starting point. In general, our idea was to have a platform where components like the service registry can be easily exchanged. Currently we are porting PeerMoS to the Android platform to experiment with P2P service overlays in realistic mobile environments also to face the challenge of heterogeneous devices.

5.1 Smooth Service Provision

To realize mobile services we have to face the challenge of smooth service provision. When migrating a service during runtime it is crucial that its provided functionality is not interrupted during this process. This is especially true in disaster recovery where an interrupted service is not only decreasing user experience but safety as well. Usually when migrating a service at runtime the hosting peer pauses the service, transfers the code and the execution state to another peer where the service is then resumed. This procedure will cause the service to interrupt for at least the time it takes to transfer the service and the execution state between peers.

A solution to this problem is to replicate the service and synchronize the execution states afterwards. This means that the service code and the execution state are copied to another peer and executed remotely. During the handover procedure state changes are propagated to the new service host using update messages. Once the services on both peers are synchronized, the service on the new peer takes over and the old service shuts down. The old service will disconnect the users only if the new service is ready to take over at the exact same state basically implementing a soft handover protocol.

In disaster recovery, voice communication is one of the most critical services. In most contemporary group voice communication software, e.g., TeamSpeak [42], a server provides the group communication functionality. The voice packets are sent to the server and then redistributed to participating clients. The main advantage to use this approach instead of directly sending voice packets to participating peers is that the server can mix all incoming voice packets into one stream resulting in a smaller number of packets with higher communication density. In terms of service overlays, we call this type of service an *interactive service*, since client peers are connected to and constantly interacting with the service. Migrating such an interactive voice service is only possible if the communication does neither stutter nor disconnect. Not only the state needs to be transferred but also the connections between clients and host. They have to be reestablished once the service is started on the new peer. Thus, all peers are first connected to both services. After the new service host sends a take-over message to all reconnected clients, the clients start to transmit their voice packets to the new host and disconnect from the old one. This introduces timing problems, because there might be clients still sending voice packets to the old host while other clients already send their packets to the new one. These packets are then propagated from the old host to the new host using update messages. Sequence numbers are introduced to allow for a seamless transmission of these delayed packets.

PeerMoS enabled us to realize a voice chat service prototype and distribute it in the P2P service overlay. To realize smooth service provision we developed persistent socket connections using two communication channels: a TCP channel for control messages and an UDP channel for data transmission. During the migration process the control message channel is paused but the data channel remains open. Data transmission then switches to the new host once the take-over message is received. First experiments conducted with our prototype are free of voice interruptions during the migration process.

5.2 Resilient Monitoring

In our framework PeerMoS, we implemented a basic monitoring scheme gathering various attributes about neighboring peers as well as invoked or connected services by exchanging simple messages. To make our monitoring protocol resilient against attacks and to cope with high mobility, heterogeneous peers, and the unreliable underlay network we are extending it with hybrid information gathering and dissemination approaches. In such resilient monitoring schemes, monitored attributes are aggregated at relatively stable peers, i.e., resource-rich peers. The information is then disseminated to all peers in the close vicinity for service registry or migration. During the data aggregation some peers may not follow the monitoring protocol and may result to Byzantine-like attacks. It may also occur that stable monitoring peers disseminate false data resulting in wrong service registry and/or wrong replication. To tackle these two situations, we focus on multi-source data aggregation and limited information dissemination. We are in the process of evaluating these ideas in our framework PeerMoS with different Byzantine strategies and various perturbation settings. We aim at providing reliable P2P service overlays and its reliability analysis in case of cascading Byzantine attacks, colluding malicious attacks, and severe perturbations.

6 Concluding Remarks

In this presented article we proposed a support infrastructure called P2P service overlay to use for *mobile crowd missions* on top of an underlay exposing *nasty characteristics*. This infrastructure is able to disseminate services in the network thus providing a wide range of functionality to participating users. We described the prerequisites of P2P service overlays as well as the main building blocks identifying major challenges arising in mobile crowd missions and especially in disaster recovery. We proposed solutions known from literature as well as own preliminary ideas to tackle some of the challenges and described our framework PeerMoS that is able to provide smooth voice chat service. Currently we are investigating approaches in the fields of matching, service placement, and service discovery. We are also working on resilient monitoring to handle Byzantine errors, potential attacks, and varying connectivity due to fluctuating demands and mobile users.

Acknowledgments

This work was funded by the German Research Foundation, Research Group 733, "QuaP2P: Quality Improvement of Peer-to-Peer Systems". This work in parts was supported by the IT R&D program of MKE/KEIT of South Korea. [10035587, Development of Social TV Service Enabler based on Next Generation IPTV Infrastructure]

References

[1] M. Lehn, T. Triebel, C. Leng, A. Buchmann, and W. Effelsberg, "Performance Evaluation of Peer-to-Peer Gaming Overlays," in *IEEE P2P*, 2010.

[2] D. Bradler, J. Kangasharju, and M. Mühlhäuser, "Evaluation of Peer-to-Peer Overlays for First Response," in *IEEE PerCom*, 2008.

[3] K. Panitzek, D. Bradler, I. Schweizer, and M. Mühlhäuser, "City Mesh - Resilient First Responder Communication," in *ISCRAM*, 2011.

[4] R. Bruno, M. Conti, and E. Gregori, "Mesh Networks: Commodity Multihop Ad Hoc Networks," *IEEE Communications Magazine*, 2005.

[5] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The Case for VM-based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, 2009.

[6] E. Koukoumidis, D. Lymberopoulos, K. Strauss, J. Liu, and D. Burger, "Pocket Cloudlets," in *ACM ASPLOS*, 2011.

[7] H. Xu and B. Li, "Egalitarian Stable Matching for VM Migration in Cloud Computing," in *IEEE INFOCOM Workshops*, 2011.

[8] K. Oikonomou, I. Stavrakakis, and A. Xydias, "Scalable Service Migration in General Topologies," in *IEEE WoWMoM*, 2008.

[9] P. Bellavista, A. Corradi, and E. Magistretti, "REDMAN: An Optimistic Replication Middleware for Read-only Resources in Dense MANETs," *Pervasive and Mobile Computing*, 2005.

[10] H. Liu, T. Roeder, K. Walsh, R. Barr, and E. G. Sirer, "Design and Implementation of a Single System Image Operating System for Ad Hoc Networks," in *ACM MobiSys*, 2005.

[11] O. Riva, T. Nadeem, C. Borcea, and L. Iftode, "Context-aware Migratory Services in Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, 2007.

[12] G. Wittenburg and J. Schiller, "A Survey of Current Directions in Service Placement in Mobile Ad-hoc Networks," *IEEE PerCom*, 2008.

[13] C. Ververidis and G. Polyzos, "Service Discovery for Mobile Ad Hoc Networks: A Survey of Issues and Techniques," *IEEE Communications Surveys & Tutorials*, 2008.

[14] "Universal Description Discovery and Integration Platform - Technical White Paper," 2000. [Online]. Available: http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

[15] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *ACM SIGCOMM*, 2001.

[16] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Middleware*, ser. LNCS, 2001.

[17] H. Pucha, S. Das, and Y. Hu, "Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks," *IEEE HOTMOBILE*, 2004.

[18] T. Zahn and J. Schiller, "MADPastry: A DHT Substrate for Practicably Sized MANETs," in *ASWN*, 2005.

[19] E. Kang, M. Kim, E. Lee, and U. Kim, "DHT-based Mobile Service Discovery Protocol for Mobile Ad Hoc Networks," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues*, ser. LNCS, 2008.

[20] D. Bradler, L. Krumov, M. Mühlhäuser, and J. Kangasharju, "Pathfinder: Efficient Lookups and Efficient Search in Peer-to-Peer Networks," in *Distributed Computing and Networking*, ser. LNCS, 2011.

[21] T. Amjad, M. Sher, and A. Daud, "A Survey of Dynamic Replication Strategies for Improving Data Availability in Data Grids," *Future Generation Computer Systems*, 2012.

[22] J. Balasubramanian, S. Tambe, C. Lu, A. Gokhale, C. Gill, and D. C. Schmidt, "Adaptive Failover for Real-time Middleware with Passive Replication," in *IEEE RTAS*, 2009.

[23] A. C. Snoeren, D. G. Andersen, and H. Balakrishnan, "Fine-grained Failover Using Connection Migration," in *USENIX USITS*, 2001.

[24] A. Fuggetta, G. Picco, and G. Vigna, "Understanding Code Mobility," *IEEE Transactions on Software Engineering*, 1998.

[25] F. Wolf, J. Balasubramanian, S. Tambe, A. Gokhale, and D. C. Schmidt, "Supporting Component-based Failover Units in Middleware for Distributed Real-time and Embedded Systems," *Journal of Systems Architecture*, 2011.

[26] K. Graffi, A. Kovacevic, S. Xiao, and R. Steinmetz, "SkyEye.KOM: An Information Management Over-Overlay for Getting the Oracle View on Structured P2P Systems," in *IEEE ICPDS*, 2008.

[27] S. C. Rhea, T. Roscoe, and J. Kubiatowicz, "Structured Peer-to-Peer Overlays Need Application-driven Benchmarks," in *IPTPS*, 2003.

[28] R. Van Renesse, K. P. Birman, and W. Vogels, "Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining," *ACM ToCS*, 2003.

[29] D. Stutzbach and R. Rejaie, "Understanding Churn in Peer-to-Peer Networks," in *ACM SIGCOMM*, 2006.

[30] S. Saroiu, K. P. Gummadi, and S. D. Gribble, "Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts," *Multimedia Systems*, 2003.

[31] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-rotaru, and H. Rubens, "Mitigating Byzantine Attacks in Ad Hoc Wireless Networks," CS, Johns Hopkins University, Tech, Tech. Rep., 2004.

[32] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based Computation of Aggregate Information," in *IEEE SFCS*, 2003.

[33] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based Aggregation in Large Dynamic Networks," *ACM ToCS*, vol. 23, 2005.

[34] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks," *SIGOPS OS Review*, 2002.

[35] P. Yalagandula and M. Dahlin, "A Scalable Distributed Information Management System," *ACM SIGCOMM CC Review*, 2004.

[36] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie, "Epidemic Information Dissemination in Distributed Systems," *Computer*, 2004.

[37] M. S. Artigas, P. García, and A. F. Skarmeta, "DECA: A Hierarchical Framework for DECentralized Aggregation in DHTs," in *DSOM*, 2006.

[38] M. L. Massie, B. N. Chun, and D. E. Culler, "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience," *Parallel Computing*, 2004.

[39] K. Albrecht, R. Arnold, M. Gahwiler, and R. Wattenhofer, "Aggregating Information in Peer-to-Peer Systems for Improved Join and Leave," in *IEEE P2P*, 2004.

[40] "Open Chord Project Homepage," 2011. [Online]. Available: http://www.sourceforge.net/projects/open-chord

[41] E. Aitenbichler, J. Kangasharju, and M. Mühlhäuser, "MundoCore: A Light-weight Infrastructure for Pervasive Computing," *Pervasive and Mobile Computing*, 2007.

[42] "Teamspeak," 2012. [Online]. Available: http://www.teamspeak.com