# RAIC Integration for Network Storages on Mobile Devices

Andriy Luntovskyy
BA Dresden University of Cooperative Education
Information Technology Department
Dresden, Germany
e-mail: Andriy.Luntovskyy@ba-dresden.de

Josef Spillner
Dresden University of Technology
Chair for Computer Networks
Dresden, Germany
e-mail: Josef.Spillner@tu-dresden.de

***Abstract*—Systems to combine multiple network and online storage targets with implied redundancy, security and fault tolerance, so-called RAICs, have recently seen renewed discussion due to the growing popularity of convenient Cloud Storage service offerings. For mobile device access to RAICs, less research results are available. A "smartphone for the future" with pervasive storage availability should be intelligently and autonomically connected to the Cloud. Such a constellation allows access without great expenses to multiple applications, data, and further resources so that the requirements of the users (security, privacy, safety, pricing, vendor selection, etc.) as well as the functional user objectives are rewarded in the best way. In addition, valuable battery capacities can be saved by selecting appropriate algorithms and parameters and by placing parts of the RAIC integration into the infrastructure. Using the example of distributed data storage, including specific resources services with versatile features such as extended storage capacity, backup, synchronization and collaborative sharing of data, we propose a mobile, energy-efficient and autonomic RAIC integration concept. Its effects are being evaluated with a storage controller on a smartphone.**

***Keywords: Mobile Cloud Access; Smartphone-for-the-Future; Network Storage Integration; RAIC; Backup Clouds.***

## I. Introduction

The term Smartphone Bloodbath has been descriptively in use in mobile phone industry reports for the race to more features and lower prices at high frequency for about three years. Essentially, a phone is technically valued by its hardware functionality and quality, its software and services ecosystem and its connectivity. Most smartphones offer sophisticated software application distribution, whereas the innovation in terms of data management is relatively slow. The separation between private and business activities reflects to some extent on data management [19], and yet most users would need a much more powerful data and storage feature set. One idea for a user-friendly "smartphone for the future" is to bind it to online storage services through a pervasive cloud of user-controlled accounts at registered providers. The online storage area allocation would grow and shrink on demand. This binding is similar to how clouds and resource-constrained cyber-physical systems and robots are already connected to each other to offload tasks from the devices into the network infrastructure [20]. One difference between phones and robots is the self-determined nature of user actions. When a user records a movie or downloads files, the phone's media size restrictions will be defused and additional functionality including online access to all private data becomes possible, although the user may decide to override the use of the online storage. The binding to multiple services at once requires intelligent client-side integration techniques with phase-of-lifecycle knowledge which additionally match the service properties against user requirements. For secure and reliable data storage, Redundant Arrays of Independent Clouds (RAIC) have been proposed as integration technique and successfully implemented for desktop computers and enterprise storage integrators [14, 15]. However, from a security and convenience perspective, on mobile devices the RAIC assembly and the distribution of the data to the attached providers needs to happen directly on the device itself in most cases, which contradicts a conservation of battery power. We therefore discuss approaches of how to integrate network storage services on mobile devices in a systematic way for predictable storage characteristics even under changing networking and device conditions.

The offered work has the following structure. In Sect. II, the basic concepts behind network and cloud storage, RAICs, and their applications, including hybrid backup clouds, are presented. The phases of the usage lifecycle of services in general and storage services in particular are examined in detail to derive a suitable integration design. Tradeoffs between user-friendly full automation and control-preserving semi-automatic or guided integration are discussed in this context.

Intelligent RAIC use in the mobile field further implies certain decisions on which algorithms, parameters and placement strategies to use in order to preserve the battery and gracefully adapt to imperfect networking conditions. Sect. III is therefore outlining specialized data coding techniques, including encryption, splitting, erasure codes and all-or-nothing transformations. Again, tradeoffs need to be understood correctly to achieve high-performance integration with low power consumption.

Sect. IV then shows the peculiarities of mobile access to RAICs using elaborated software architecture on a selected smartphone platform. The implementation is introduced and first results are presented.

Finally, the work is concluded with a summary of the findings and an outlook on further ideas to improve the connections of smartphones into the cloud.

## II. Background on Storage Service Integration

Cloud storage is often used for backups, but also for extended storage capacity and sharing of data between devices and users. Up-to-date cloud technologies aimed to backup and restore routines of critical enterprise or authority data are discussed in [7–10]. A scheduled comparative analysis of existing complex solutions and standalone tools has been done and represents the advantages of combined (private + public) clouds regarding to traditional data-center backups and some known cloud backup solutions [1–6].

In order to achieve full convenience and elasticity, clients require an intelligent combination of externally maintained public storage clouds with use of efficient cryptographic methods and stripes/parity dispersal functionality for authenticated, transparently encrypted, low-overhead and reliable data access. This approach has become popular with the name RAIC – Redundant Arrays of Independent Clouds [6] in analogy to RAID.

One RAIC realization is the deployment of the hybrid clouds as a combination of private and public clouds in certain topologies. The combined hybrid clouds with additional cryptographic protection functionality and management layer (so called “cloud storage controller”) at the client side is often an appropriate solution [5, 19, 20]. Taken to the extreme, such setups can include peripheral devices such as USB sticks for a four-eye principle in access control.

Further key points of a hybrid cloud backup concept under the given circumstances are as follows:

1. Use of effective and transparent encryption methods (for instance, ***RSA+AES+PKI***, see Fig. 6) for increased security.
2. Deployment of a stripe and parity based dispersion (analogically to the known ***RAID*** techniques, see Fig. 1) for increased safety.

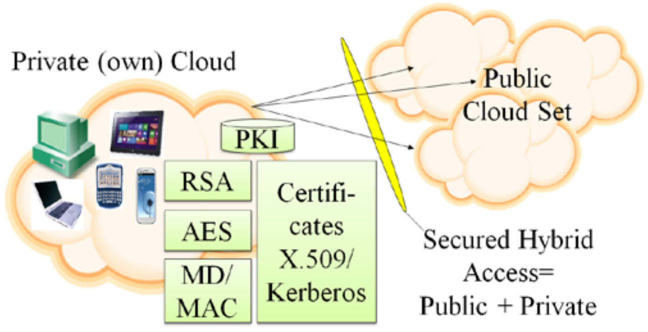

Figure 1. Cloud Backup and Transparent Encryption: (MD) Message Digest; (MAC) Message Identification Code; (AES) Advanced Encryption Standard; (RSA) Rivest, Shamir and Adleman Encryption; (PKI) Public Key Infrastructure (X.509/ Kerberos).

The notion of transparent encryption for cloud backup encompasses the following features (cf. Fig. 1):

1. Efficient cryptography methods: ***AES, RSA, MD/MAC***.
2. ***X.509/Kerberos*** private keys, ***PKI*** deployment.
3. Document classification and demarcation.
4. Analysis of structured, unstructured data and context information.
5. User authentication and respective keys granting.

Many RAIC characteristics can be explained with corresponding RAID methods and literature. In local backup setups, the most popular systems are the RAIDs numbered as 0, 1, and 5, correspondingly with two or four disks of which zero or one are redundant.

The functionality of RAIDs is based on stripes and parity dispersal routines [18]. In Fig. 2 for a RAID5 a representation is depicted. With different colors the partition in the usual disks array is given: firstly for the data (the so called “***Stripe Set***”, e.g. A1 or C3) and then the distribution of the parity sums (“***Parity Set***”, e. g. BP or DQ) through the five disks Disk 0 … Disk 4. In the given case the common available volume ***V*** for the data backup will be calculated with the formula (cp. Fig. 2c):

$$V=(n-1)\cdot V_{m\ i} \qquad (1)$$

where ***n*** – number of used HDDs, ***Vmin*** – minimal available HDD volume in the array. The redundancy is self-evident preconditioned via the parity set.

***<u>Example 1 (RAID efficiency)</u>.*** Let us here consider the example with four arrays each of 500GByte for RAID5:

$$V=(4-1)\cdot 5\quad G\quad 0B= \qquad (2)$$

***1500 GB pure for data backup as well as 500 GB for the parity control*** (see Fig. 2c).

Therefore a next constructive idea is the deployment of redundant cloud arrays (stripe and parity based dispersion). There are naturally a lot of further ***RAID*** concepts optimized for: minimum access time, minimum failure probability, maximum volumes, minimum costs [9, 18].

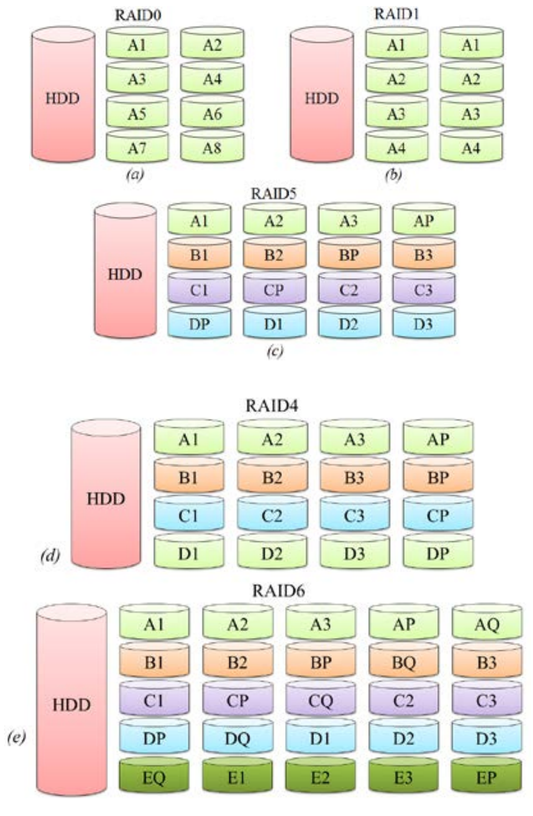

Figure 2. The most used systems: RAID 0, 1, 4, 5, 6: (RAID) Redundant Array of Independent Disks; (HDD) Hard Disk Drives (up to five disks, Disk 0 … Disk 4).

Practically, these multiple RAID concepts can be continued and mapped to RAICs. There are already numerous subconcepts of ***RAICs***, or ***Redundant Arrays of Independent Clouds*** [1, 10, 19, 20]. Existing variations to the concept are:

- RAINS – Redundant Array of Independent Networked Storages;
- RAIDC – Random Array of Independent Data Centers;
- RAOC – Redundant Array of Optimal Clouds, an extension to RAIC which emphasizes an enforcement of user requirements on the selection and maintenance of storage service arrays.

***RAID DP (Double Parity)*** is a block-level RAID system with double striping of parity information on separated HDDs based on both RAID4 and RAID6 structures. The second parity Q (see Fig. 3) can be computed with the same formula as the first parity P but with other data stripes [23].

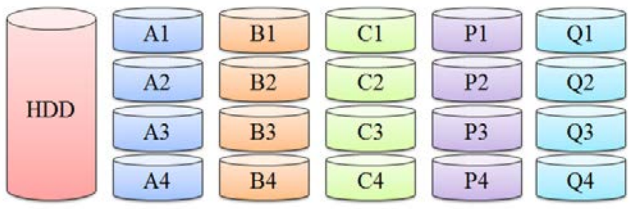

Figure 3. RAID Double Parity Structure.

The first parity is horizontal, the calculated second parity Q diagonal, see formula (3):

$$\begin{aligned} P1 &= XOR(A1, B1, C1)\\ P2 &= XOR(A2, B2, C2)\\ P3 &= XOR(A3, B3, C3)\\ P4 &= XOR(A4, B4, C4) \end{aligned} \tag{3}$$

$$\begin{aligned} Q1 &= XOR(P1, A2, B3, 0)\\ Q2 &= XOR(P2, A3, 0, C1)\\ Q3 &= XOR(P3, 0, B1, C2)\\ Q4 &= XOR(0, A1, B2, C3) \end{aligned}$$

Since in a RAID DP any two disk failures can be compensated the availability of such a system is increased compared to a single-parity solution. The recommended RAID-DP sets consist usually of 14 + 2 HDDs. The restoring via RAID DP is relatively simple.

The further advantages of RAID DP are the simplicity of XOR-Operation for parity computing and possibility to conversion to RAID 4 via switching-off of the Q-stripes.

Deployment of optimized RAIC DP offers the advantages as follows:

$$n \geq 5, Netto/Brutto \geq \frac{n-2}{n}, \quad Failure_{Security} = 2 \tag{4}$$

in comparison to well-known RAIC5 (cp. Fig. 2c).

All services offered over the Internet are interacted with according to certain usage lifecycle phases. Storage services are no exception, they also adhere to a lifecycle. Fig. 4 presents the relevant phases and introduces suitable client-side integration handlers for each phase. The first three phases (discovery and selection, contracting and configuration) can be subsumed under the term matchmaking. These phases typically apply once per user-service relationship. The fourth phase, usage, is executed more than once and depends on the preceding phases.

The presented service integration concept is a general one. For mobile clients bound to storage services in the cloud, its interpretation is as follows. During the service discovery, a dialogue on the device screen guides the user to the right storage services for any given situation. By using automation and autonomic computing concepts, the dialogue can be kept simple or even not be shown at all, at the expense of honoring custom user preferences. Then, more client-side agents perform the necessary configuration of the services, including account creation and registration within the storage controller. Finally, a scheduler within the storage controller orders the timely transmission of data to and from the device.

Agent frameworks to handle the sign-up to services already exist, for example OSST, the Online Service Sign-up Tool [20]. The frameworks assume access to a well-maintained service registry which not only contains information about the services, but also links to service-specific agent extensions. However, the frameworks need to be implicitly parameterized according to the specific needs of mobile users and with appropriate information already present on the mobile device, including identities (Fig. 4).

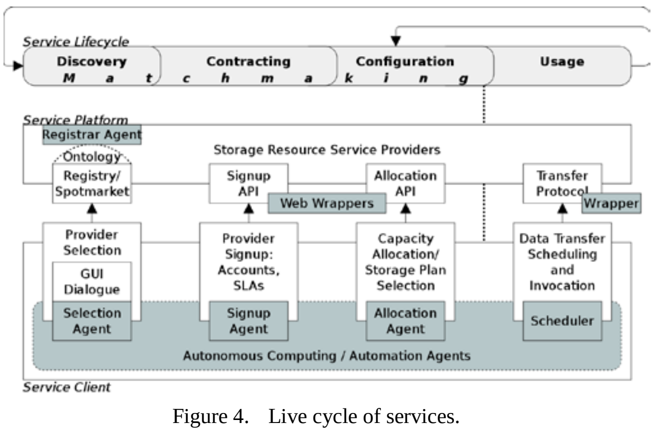

Figure 4. Live cycle of services.

In summary, the presented background information demonstrates the feasibility of integrating storage services on mobile devices in a partially automated process. The next section will give detailed insight into appropriate choices of methods and their parameterization.

## III. Efficient Access to Storage Services from Mobile Devices

Depending on the use cases, the weight of comparison parameters to distinguish the most suitable RAIC integration method differs. For many client systems, security plays a major role and motivates distributed data storage with comparatively more storage overhead in return for higher security. As a generalization thereof, subjectively optimal parameters including storage and retrieval times and service costs can be considered and weighted by clients at configuration time, yielding optimally distributed storage or RAOCs [14]. For mobile devices, two parameters become dominant in the RAOC scheme: The energy efficiency of the integrating device and the usability under imperfect networking conditions. Both have so far not been subject to analysis for the research on RAICs, but are crucial for the further acceptance of such techniques.

Energy consumption can be broken down into the (negligible) setup, service selection, signup and configuration/reconfiguration processes, which typically don't happen more than once per device power-on session, and the service usage processes for storing and retrieving data. Measuring the energy efficiency of algorithms requires specialized equipment. The electrical power consumption is not linear to the performance, but grows along with it, hence a performance comparison assuming equal processor load can be used for a first estimation. For a detailed power consumption analysis, we make use of the HAEC - Highly Adaptive Energy-Efficient Computing measurement infrastructure, as shown in the photo below (Fig. 5). The devices are Yokogawa WT210 digital power meters which measure both the incoming AC and the resulting DC consumed by the processor.

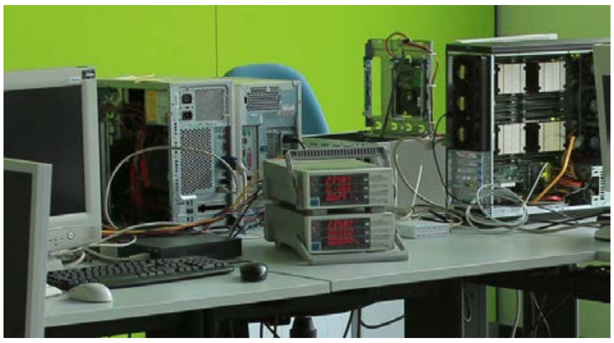

Figure 5. HAEC laboratory measurement equipment.

Calculated performance characteristics of RAIC integration techniques based on [22] are summarized in Table I for read and write access. The energy efficiency can however not be calculated easily and must be measured per technique.

TABLE I. QUALITATIVE COMPARISON OF PERFORMANCE CHARACTERISTICS FOR VERSATILE RAIC INTEGRATION TECHNIQUES

| Technique | Read performance | Write performance |
|---|---|---|
| RS erasure code, 0% redundancy, XOR | 100% | 100% |
| RS erasure code, 0% redundancy, SIMD | 270%-1200% | 270%-1200% |
| RS erasure code, 50% redundancy, n=3 | 100% | 67% |
| AONT-RS, n=3 | 33% | 33% |

Note: SIMD – Single Instruction, Multiple Data computing structure by M. Flynn (1970); RS – Reed-Solomon-Code (deployed since 1977); AONT – All-or-Nothing-Transform by encryption by R. Rivest (1997)

In Fig. 6 we show the results of measuring the AC when running a cloud storage controller testsuite which systematically varies storage selections and redundancy parameters. The results are preliminary and do not yet allow for a mapping of parameters to power spikes but they clearly show variation levels which warrant research on energy efficiency optimization in the future.

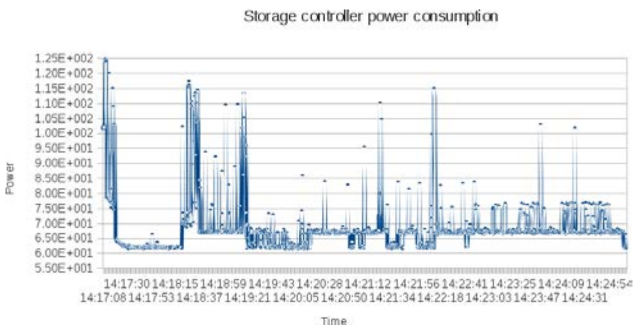

Figure 6. Storage controller power consumption over time.

Imperfect networking usability mandates an intelligent use of caching and scheduling so that slow or broken links will show no or little effect on the user of a RAIC. This typically differs per implementation. However, already on the algorithmic level, some erasure codes have been more optimized for storage, retrieval and repair than others. Researchers have identified suitable algorithms through experiments [22]. Based on these observations, we can assume that the use of processor-specific erasure codes is beneficial for mobile devices.

Both the device's energy efficiency and the imperfect networking usability can be tremendously improved by placing the RAIC integration onto a trusted local network proxy. So-called storage integrators or proxies can serve multiple users and enforce group policies. On the other hand, they have drawbacks concerning the trust, mobility and overall distributed system’s energy efficiency given that such additional devices will remain idle for long durations. Fig. 7 shows both possible integration approaches in a comparison architecture scheme.

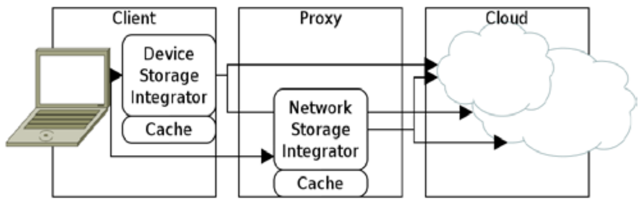

Figure 7. Variants for efficient placement of RAIC Integrator between the clouds.

## IV. A NEW MUST-HAVE APP: RAIC INTEGRATOR FOR SMARTPHONES

While our results are generally applicable to all mobile devices including tablets and notebooks, our realization scenario focuses on mobile phones due to their increasing popularity as “Swiss Army Knives” for computing tasks. Today, such phones ship with internal storage media (ROM, non-volatile flash memory, SD cards) and otherwise rely on manual storage service integration beyond the sometimes preconfigured vendor-specific services. Increasing amounts of data produced by mobile phone sensors and applications push the idea of a "smartphone for the future" with ubiquitous access to elastic storage in the cloud. Such a smartphone requires inter alia an operating-system integrated library for transparent RAIC integration across all applications which need extended storage capacity, offsite backups and other uses of storage. Essential parts of the integrator are (1) a database with information about available services, including their functional and non-functional properties and protocols for accessing them, (2) protocol-specific access modules, (3) a dispersion module which splits the data according to the user-defined parameters while considering energy efficiency and imperfect networking conditions, and (4) autonomic support functions for service sign-up and repair in case of failures.

The binding of a mobile phone to a RAIC-DP configuration through an integrator is depicted in Fig. 8. The P-stripe is stored in the private cloud client, while the Q-stripe is delegated to the public clouds, i.e. to the provider. Arbitrary RAIC and dispersion configurations are possible, although certain key configurations will be preferred by mobile users: RAIC-DP for highest safety, AONT for highest (information-theoretic) security, and JBOC/RAIC0 for the least amount of overhead. A configuration wizard would have to present these choices to the users in a meaningful way.

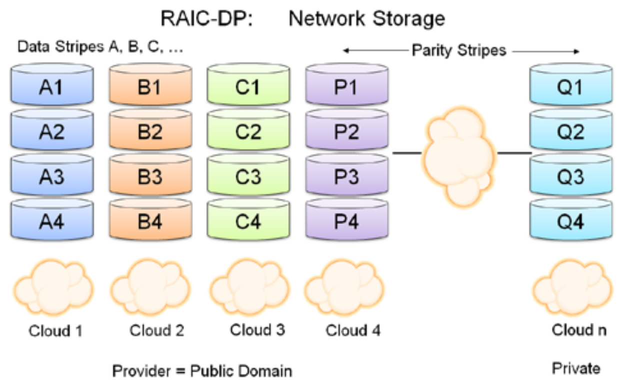

Figure 8. RAIC-DP: A Network Storage Model.

A suitable software architecture for the realization of a mobile RAIC over both local and cloud storage resources is depicted via Fig. 9, following the design proposed for generic cloud storage controllers [14]. The predominant client-side software for RAICs consists of the following three layers with the related functionality:

1. Integration layer: logical partition and interface to the backup application.
2. Pre-processing layer: stripes/parity dispersal routine, encryption and other modifications.
3. Transport layer: block transfer.

The clients obtain the possibility of reliable and efficient access to an array of virtualized storage media, offered as a service or as local complementary media, with added organizational and spatial independence. This software considers the state-of-the-art.

The offered software layered architecture realizes a RAIC concept [10, 19 – 21] and includes the following already known components with the extended functionality:

1. Advanced integration layer: A local virtual file system interface available to all applications. Depending on the operating system, there may be additional specific interfaces, for instance the registration as content provider on Android or the export as RESTful web service through RestFS.
2. Advanced pre-processing layer: Codecs: classification of document types and coding (text files, MPEG, PDF); Policies on the data storage subjects and paths; Stripes/parity dispersion routines; Authentication with MD/RSA/PKI; Encryption with AES/RSA/PKI.
3. Advanced transport layer: Parallel and block-wise streaming; Caching and local persistence; Adapters for multiple provider APIs.

The proposed system has been implemented with existing academic and open source software for a proof-of-concept prototype. Nubisave [14] is a cloud storage controller which performs the functionality of the upper layer as a Linux user-space file system (FUSE) module with *1* file input and *n* fragment outputs. Through the Nubisave configuration GUI, the remaining two layers can also be controlled. For instance, the Nubisave splitter module's first output can be connected to an EncFS module for data encryption, which is in turn connected to a FuseDAV module for placing the encrypted fragment data on a protected WebDAV folder which serves as standard-compliant interface to a cloud storage area.

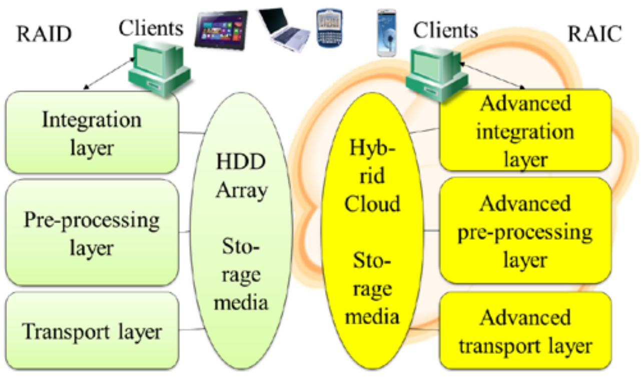

Figure 9. Offered software architecture to realization of a RAIC: (HDD) Hard Disk Drive, or other local drives including SD media; (RAIC) Redundant Arrays of Independent Clouds.

Some mobile phone operating systems run directly on Linux, including Maemo and the more recent SailfishOS and FirefoxOS, so that Nubisave's file system interface is a suitable means for data access across all applications. For Android and similar systems with restricted global data access, a translator between files and the respective per-application content API would be required, which we haven't created yet.

Imperfect network handling is an implementation detail of the transport modules. We have previously refined fault-tolerance access to RESTful services (including e.g. WebDAV as HTTP extension) as RAFT-REST concept. The Java ResUp library [21] is available to transport module authors as a convenient caching and retransmission handler. For the proof-of-concept prototype, we have integrated it into a module to access a RESTful S3-like blob storage service. Beyond the specific transport modules, Nubisave also caches data by itself to some extent.

Hence, the combination of a cloud storage controller with energy-efficient parameterization, agent-based service lifecycle handling for semi-automatic integration and fault-tolerant service integration under imperfect networking conditions is possible today and fulfils the requirements set forth in the introduction of this paper.

## V. Conclusion

This paper has given a brief systematic introduction into the challenges of integrating redundant arrays of network and cloud storage services into mobile devices.

Sect. II introduced background topics, namely RAICs for distributed storage and storage service lifecycle handling, before Sect. III highlighted mobile-specific integration concerns. The software realization description and discussion in Sect. IV completed the approach by focusing on the already achieved results in contrast to the still open questions.

For smartphone makers, the results show that ubiquitous storage integration is a desirable feature which leads to outstanding devices with a functionality closer to what highly demanding users expect.

## References

[1] C. Baun, M. Kunze, J. Nimis, S. Tai. Cloud Computing. Web-based dynamic IT-Services, Springer Verlag, 2010 (in German).

[2] Amazon Web Services (Online 2013): http://aws.amazon.com/.

[3] Google Drive (Online 2013): https://drive.google.com/.

[4] Shelton Shugar. Cloud Computing at Yahoo, SVP Yahoo Inc. (Online 2013): http://opencirrus.org/.

[5] H. Kommalapati. Windows Azure Platform for Enterprises (Online 2013): http://msdn.microsoft.com/en-us/magazine/ee309870.aspx/.

[6] D. Decasper, A. Samuels, J. Stone. RAIC – Redundant Array of Independent Clouds// Patent USA 2012: Reg. No.: 12/860, 810, Publishing No.: US 2012/0047339 A1.

[7] A. Luntovskyy. Programming Technologies of Distributed Applications, Kiev DUIKT University of Telecommunications, 2010, 474 p. (ISBN 978-966-2970-51-7, monograph in Ukrainian).

[8] A. Luntovskyy, D. Guetter, I. Melnyk. Planung und Optimierung von Rechnernetzen: Methoden, Modelle, Tools für Entwurf, Diagnose und Management im Lebenszyklus von drahtgebundenen und drahtlosen Rechnernetzen, Vieweg + Teubner Verlag Wiesbaden / Springer Fachmedien Wiesbaden GmbH, 2011, 411 p. (ISBN 978-3-8348-1458-6, in German) // Handbook for German universities.

[9] A. Luntovskyy, M. Klymash, A. Semenko. Distributed Services for Telecommunication Networks: Ubiquitous Computing and Cloud Technologies, Lvivska Politechnika, Lviv, 2012, 368 p. (ISBN: 978-966-2405-87-3, monograph in Ukrainian).

[10] A.Luntovskyy, V.Vasyutynskyy, J.Spillner. RAICs as Advanced Cloud Backup Technology in Telecommunication Networks. International Research Journal Telecommunication Sciences, Kiev, ISSN 2219-9454, Telecommunication Sciences, 2012, Volume 4, Number 2, pp. 30-38.

[11] VMware vSphere API for Storage Awareness (Online 2013): http://www.vmware.com/.

[12] Citrix Systems: ShareFile (Online 2013): http://www.citrix.com/products/sharefile/overview.html.

[13] Sheikh M. Habib, S. Hauke, S. Ries, and Max Mühlhäuser: Trust as a Facilitator in Cloud Computing: A Survey. Journal of Cloud Computing: Advances, Systems and Applications, SpringerOpen, June 2012.

[14] J. Spillner, G. Bombach, S. Matthischke, R. Tzschicholz, and A. Schill: Information Dispersion over Redundant Arrays of Optimal Cloud Storage for Desktop Users. In: IEEE International Conference on Utility and Cloud Computing. Melbourne, Australia, pp. 1-8, December 2011.

[15] R. Seiger, S. Gross, and A. Schill: A Secure Cloud Storage Integrator for Enterprises. International Workshop on Clouds for Enterprises, pp. 252-255. Luxembourg, September 2011.

[16] Johannes Schad, Stephan Zepezauer, Josef Spillner: Personal Cloud Management Cockpit with Social or Market-Driven Asset Exchange. Networked Systems (NetSys/KiVS), Stuttgart, Germany, March 2013.

[17] J. Spillner, A. Schill. A Versatile and Scalable Everything-as-a-Service Registry and Discovery. Proc. 3rd International Conf. on Cloud Computing and Service Science (CLOSER), pp. 175-183, Aachen, Germany, May 2013.

[18] A. Schill, J. Spillner, S. Gross. FlexCloud@TUD Project/ Dresden University of Technology TUD (Online 2013): http://www.rn.inf.tu-dresden.de/

[19] H. Kim, N. Agrawal, C. Ungureanu: Revisiting Storage for Smartphones. ACM Transactions on Storage 8(4), November 2012.

[20] J. Spillner, C. Piechnick, C. Wilke, U. Aßmann, A. Schill: Autonomous Participation in Cloud Services. Proc. 2nd ITAAC Workshop, pp. 289-294, November 2012.

[21] J. Spillner, A. Utlik, T. Springer, A. Schill: RAFT-REST - A Client-side Framework for Reliable, Adaptive and Fault-Tolerant RESTful Service Consumption, ESOCC 2013, The European Conference on Service-Oriented and Cloud Computing, Malaga, Spain, Sept. 2013.

[22] J.S. Plank, K.M. Greenan, E.L. Miller: Screaming Fast Galois Field Arithmentic Using Intel SIMD Instructions. Usenix FAST, February 2013.

[23] Proceedings of the Third USENIX Conference on File and Storage Technologies (Online 2013): http://static.usenix.org/publications/.