Fault Tolerance Support of Smart-M3 Application on the Software Infrastructure Level

Ivan V. Galov* Dmitry Korzun*†
*Petrozavodsk State University (PetrSU)
Petrozavodsk, Russia
†Helsinki Institute for Information Technology (HIIT)
and Department of Computer Science and Engineering (CSE), Aalto University
Helsinki, Finland
{galov, dkorzun}@cs.karelia.ru

***Abstract*—The Smart-M3 platform allows constructing software applications consisting of agents that interact by sharing information in a smart space. An important problem is achieving the dependability of applications in case of failures, a common situation for existing networked environments. The most sensitive operation is subscription—a persistent networked query. In this paper, we consider a generic software infrastructure for a Smart-M3 application and propose infrastructure-level solutions to support the application fault tolerance. Our first solution augments the infrastructure with a content service. It improves safety (preservation) and integrity of volumetric data due to the delegation of storage functions to a separate element of the application infrastructure. The service employs the well-known matured web technology. The second solution consists of mechanisms for the fault tolerance support to detect failures of subscription and to recover effectively. For case study, we use an existing Smart-M3 application—the SmartRoom system. Our evaluation confirms that our solutions lead to better dependability without essential degradation of the performance.**

I. Introduction

Smart space concept relates to creating of ubiquitous computing environments. Participants of this environment are involved in collaborative accumulation and sharing of information about the environment and participants [1], [2], [3], [4]. Every smart space forms service-oriented integrated system that provides: 1) service composition based on dynamic participants interaction via accumulated information sharing and 2) service delivery to end-user via various interface devices of a computing environment. Typically, a computing environment is localized in a physical spatial-constrained space with many locally equipped devices. In particular, Internet of Things (IoT) technologies are utilized for organization of network communication between devices and external systems (e.g., web services in the Internet).

Smart-M3 platform [5] is an open source research prototype intended for creating smart spaces in IoT environments. Every Smart-M3 application is built up as a distributed system of software agents interacting with each other and running on computing devices of the IoT environment. In Smart-M3, an agent is called knowledge processor (KP). They interact and share information with each other through a specialized subsystem (server)—semantic information broker (SIB). It controls access to shared information in the smart space. Such indirect interaction follows the well-known blackboard [6] and publish/subscribe [7] models. Changes made by one KP are available to other KPs [8]. The subscription operation allows any KP to perform persistent queries to keep a track of information changes in the smart space [5], [9]. One or several KPs construct a service, which, in turn, is delivered to other services or users (via their client KPs).

Dependability of Smart-M3 applications is disrupted by frequent failures of the IoT environment or elements of the application itself. Generally, fault tolerance constitutes in ability of the application to deliver its service in presence of faults. In Smart-M3, the subscription operation is one of the most sensitive operations for faults. Such issues as a network disconnection or loss of regular subscription indications lead to essential failures that breaks the application dependability. For a Smart-M3 application, we distinguish its software infrastructure, which provides the means for application operation. Our focus is on solutions implementable on the infrastructure level.

There is a number of works, including [10], [11], [12], [13], that studied an application infrastructure for smart spaces and ubiquitous computing environments. They considered either very abstract infrastructures or very particular ones. These approaches cannot be applied directly to Smart-M3 applications. In this paper, we analyze generic software infrastructures for Smart-M3 applications in respect to the problem of fault tolerance. We evolve the notion of the Smart-M3 application infrastructure, which previously appeared in [14], [15], [16].

For the fault tolerance support in a Smart-M3 application, we proposed two solutions. The first one augments the infrastructure with a special service to support an operation with volumetric data. Introduction of an application-wide file storage is a well-known approach, which is also beneficial for smart spaces. This service preserves collected data files and manipulation with them in case of failures. The second solution includes two mechanisms. 1) Subscription control allows to determine problems with the subscription whether it was disrupted or some data were lost. 2) Fault recovery allows an application to restart and continue its services if failures are detected (automatically or manually).

ISSN 2305-7254

The rest of the paper is organized as follows. Section II describes related work. Section III considers a generic software infrastructure applicable for Smart-M3 applications. Section IV introduces a new element to infrastructure—Content-service, which is used for enhancing data integrity in the application. Section V presents several mechanisms to improve the fault tolerance of Smart-M3 applications. Section VI evaluates the proposed solutions. Section VII summarizes the contribution of this paper.

II. Related work

The concepts of middleware platforms and software infrastructures for applications in ubiquitous computing environments are well-known. SISS [10] is an example of many existing platforms for creating applications based on agent interactions. In particular, SISS defines a layered software infrastructure: the communication layer provides QoS and reliability in agents interaction, the coordination layer supports agent collaboration, and the service layer contains common shared services.

Wang *et al.* [11] proposed a class of infrastructures for smart spaces based on Semantic Web technologies. Such infrastructures are focused on explicit representation, expressive querying, and flexible reasoning of contexts.

Sathish and di Flora [12] introduced a infrastructure development framework for smart spaces. It aims at infrastructures with dynamic service compositions and takes into account security and privacy issues. An infrastructure consists of several modules such as modules for data representation and access, data storage and manager, external data repository, and support of data security and privacy.

da Costa *et al.* [13] considered reference infrastructures for ubiquitous computing and formulated their key development issues and challenges. An architectural model was proposed, where the dependability becomes a key property, along with the heterogeneity, scalability, and interoperability. A notable conclusion is that failure-detection and recovery strategies, such as check-pointing, compensation, isolation, or reconfiguration, need to be applied in ubiquitous computing.

The above work provides characteristic properties, reference models, and case studies for development of application infrastructures in smart spaces. In contrast, we consider the case of Smart-M3 applications and propose particular solutions applicable for this case.

In the Smart-M3 case, Vasilev *et al.* [17] proposed a substitution mechanism, which allows substituting a compromised agent with another one. The application dependability is improved on the level of software infrastructure. Replacement agents are additional infrastructural elements, thus making development and deployment more complicated. In contrast, our solutions do not rely on extensive expansion of the infrastructure with many additional agents.

III. Infrastructure of Smart-M3 application

Software infrastructure of a Smart-M3 application consists of the SIB and those KPs (infrastructural KPs) that are responsible for service construction and delivery [14], see Fig. 1. Infrastructure is deployed in computing environments including the host devices, network equipment, and system software, which provide operation and network communication. In general, deployment is a process of installing, setting up, and launching all infrastructural elements. We consider three deployment options for infrastructural KPs, see Table I.

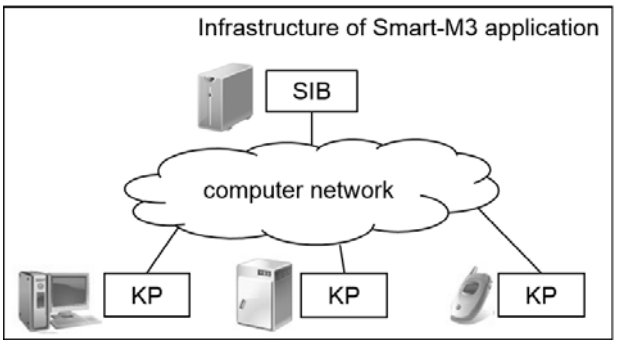

Fig. 1. Smart-M3 application infrastructure consists of the deployed SIB and service-responsible KPs

TABLE I. Deployment options for infrastructural KPs

Option	Properties
1. Clustering near the SIB	Effective access to information shared in the smart space. Online 24/7 service mode with easy installation and maintenance. Example: core services of application.
2. Device-aware location	Effective interaction with involved service-specific devices. Example: KPs for sensors, cameras, climatic equipment.
3. Server-oriented location	Extensive or complex data processing. Example: mediation of external data sources and services.

The first deployment option assumes launching a KP on the same machine where the SIB operates. Typically, the SIB is continuously running on a powerful server computer, so the KP is able to run persistently, to perform resource-intensive operations, and access the smart space with no network latency.

The second deployment option is running a KP on a dedicated computer connected to one or several specific devices which are required for service delivery (projector, interactive whiteboard, sensor, camera, microphone, etc.). For instance, a KP that supports the participation of low-capacity devices in the smart space, since they are not able to run the KP themselves [18].

The third option is launching a KP on a server machine different from the one with the SIB running. Such KPs usually deliver services that require resource-intensive computations and can affect on the SIB performance. In particular, a mediator KP connects external data sources with the smart space and performs data transformation and synchronization [19]. This deployment option assumes a variety of computers, from local and corporate servers to cloud systems in the Internet.

Consider SmartRoom system [15], [16] for an example of software infrastructure. The system is used for automated holding of collaborative activities such as conferences, meetings, or lectures. The basic installation for conference mode includes KPs responsible for the following services: conference management (Conference-service), presentation control for speaker (Presentation-service), and storage of participants' materials for use during presentations (Content-service). The deployment scheme is shown in Fig. 2.

Computing environment for SmartRoom is localized in a

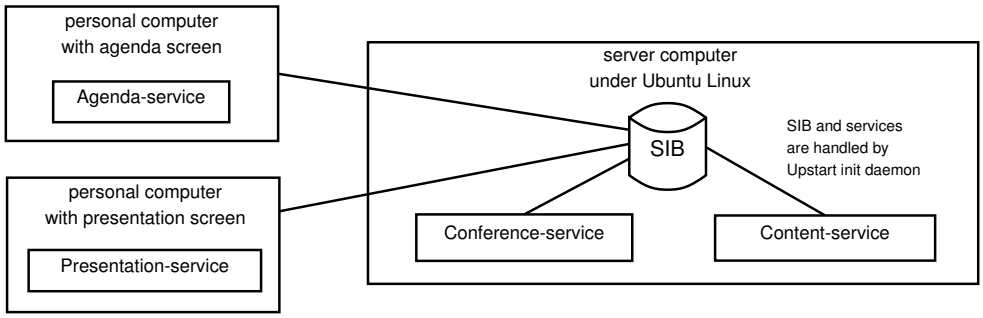

Fig. 2. Deployment options in the SmartRoom infrastructure: Conference-service and Content-service are near the SIB, Agenda-service and Presentation-service are near their multimedia screens

physical room equipped with various computing and media devices. Devices are connected via wireless and wired local area networks. There are two public screens in the room: for presentation slides and for activity agenda. Two projectors are assigned with these screens and connected to two personal computers. The server machine running the SIB can be either physically presented in the room or remote (e.g., in a corporate network). The KPs of Conference-service and Content-service are launched on the server machine with the SIB to reduce network latency (1st deployment option in Table I). The KPs of Agenda-service and Presentation-service operate on computers connected to the projectors (2nd deployment option).

In general, a software infrastructure provides means for application operation and ensures the application is operating in a proper way. From this vision, a key characteristic is dependability. A dependable application is such that [20] "reliance can justifiably be placed on the service it delivers". Any infrastructure must provide the following common attributes of the application dependability [20].

Availability: readiness for usage.

Reliability: continuity of service.

Safety: nonoccurrence of catastrophic consequences on the environment.

Confidentiality: nonuccurrence of unauthorized access.

Integrity: nonoccurrence of improper alterations of information.

Maintainability: ability to undergo repairs and evolutions.

Runtime failures decrease the application dependability. One of the approaches to resist failures is introduction of support for fault tolerance. In general meaning, fault tolerance includes methods to ensure that the application is capable to deliver its services in the presence of faults.

At the moment, a Smart-M3 application suffers from the lack of reliability and integrity, which clearly leads to reduced availability of the service provision. Failures in basic infrastructural elements or during the subscription operation result in improper application behavior up to breaks and complete termination of the service delivery. Furthermore, data integrity can be violated, thus leading to data corruption or loss.

We distinguish three failure reasons, which affects on the Smart-M3 application dependability: 1) SIB failures, 2) WiFi network failures, 3) infrastructural KP failure. Table II provides their summary.

TABLE II. INFRASTRUCTURE ELEMENTS FAILURE REASONS

Element	Failure
SIB	software error (freezing, crashing); lost subscription connection with KPs; data loss (overload).
Wireless network	subscription connection breaks; data packets loss.
Infrastructural KP	lost network connection with the SIB; software error (crashing).

SIB failures are most serious as they impact on all KPs which interacts with the SIB and reduce the overall reliability. Such failures are related to errors in current SIB implementation (prototype status) as well as the SIB overload caused by frequent queries to shared information of the smart space from multiple parallel KPs and large volumes of processed information. Wireless network failures correlates with the network overload, which leads to high latency or data loss. These problems can be bypassed with powerful Wi-Fi equipment and additional application-level mechanisms for control of data loss. Failures in a particular KP are also related to errors in the program code or happen due to dynamic nature of KP participation in the smart space.

An unsafe point, which highly impacts the applications reliability, is the subscription operation. Failures in subscription lead to loss of subscription notifications from the SIB to its KPs (indication that subscribed information has been changed). In particular, network connection disruption for subscription requires the KP to establish a new subscription (resubscription).

To support the application fault tolerance we propose two solutions intended to improve reliability and data integrity of Smart-M3 applications. The first solution is addition of a content service, as one of basic infrastructural elements, to the software infrastructure of a Smart-M3 application. This service supports volumetric data integrity and safety upon the occurrence of failures, see Section IV. The second solution consists of mechanisms for subscription control and service restart. They aim at determining problems with subscription and subsequent recovering, see Section V.

IV. CONTENT-SERVICE

Many Smart-M3 applications have to deal with volumetric factual data, e.g., see [19], [4]. Data are collected from

external sources (web services in the Internet, data bases, cloud storages) as well as generated or accumulated during application operation due to activity of the participants. In the considered example of the SmartRoom system, such data include presentation, audio and video content, activity history, or traces of happened events.

Every service produces and/or processes information, which, in turn, is used by other services or users. Conceptually, a smart space should not substitute existing data storages. Instead, it acts as a semantic information hub that links heterogeneous data sources and fragments of knowledge with each other [3], [4], [19]. Factual data, unlike dynamically changing semantic links between them, should be kept in specialized storages that provide more effective domain-specific functions for data storing, accessing, and retrieval.

We propose adding a dedicated service—Content-service—to the Smart-M3 application infrastructure. The service role is persistent storage of volumetric data. What data to store is determined by the application itself, and any its KP can exploit the service to store data as files. The service employs well-known file sharing web technologies, which are mature and considered as providing high dependability. Every KP (or user through a web interface) can upload some data (text, images, audio, video, etc.) to the service and then share a download link for the file. Files can be uploaded directly from KPs or users as well as from external sources (e.g., video). Data from external sources are duplicated in the service, allowing reduction the data access time as the service is located closer to the users. In this case, Content-service acts as a cache. Another advantage of the service is support of portable applications working in the LAN. It helps to use and accumulate data in the environments without the Internet access.

The proposed architecture of Content-service is shown in Fig. 3. The service is a web application. In our proof-of-the-concept implementation for SmartRoom[1], the service is written in Python. Apache web server is used for hosting. Interaction between the web server and the web application is performed via a web server gateway interface (WSGI). Content-service consists of three modules: logic, HTML templates and KP. The logic processes HTTP requests received from the web server, generates responses using HTML templates (templates of web application pages), saves received files (user's content). The KP implements interaction with the smart space, where links to files are published for shared use. Also, the KP provides necessary information from the smart space to the logic (user authentication check, already shared links). Apache web server allows other KPs to receive (request for a file) stored files directly using a published link from the smart space.

The process of file uploading and sharing is described in Fig. 4. User 1 passes web authentication and uploads necessary file(s) to the service via a web form. The file is sent to the web server using the HTTP protocol. Then the file is passed for processing to the Content-service logic. The service saves the file in a special directory of the file system. All files in this directory are shared and can be accessed by HTTP requests to the web server. Finally, the service publishes a sharing link to file in the smart space. If user 2 reads the file link from the

[1]The open source code (GPLv2) is available at sourceforge.net/projects/smartroom/files/services/content-service/

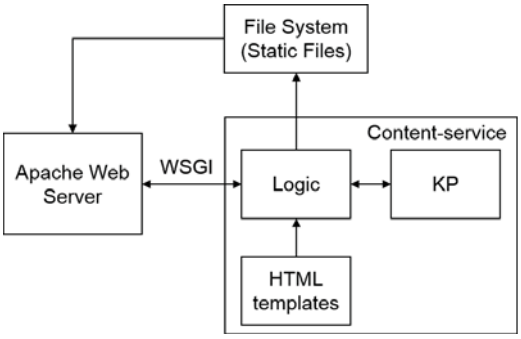

Fig. 3. Content-service architecture: the use of matured web technologies to improve the fault tolerance

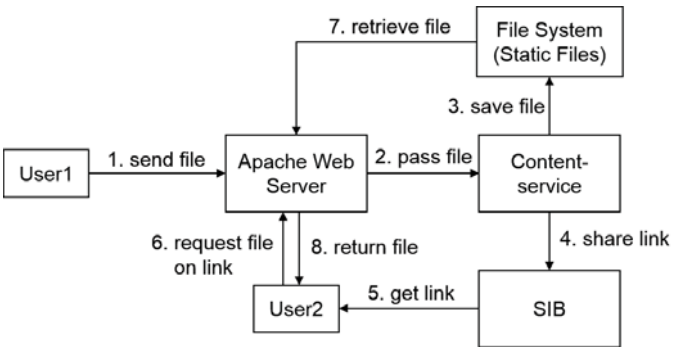

Fig. 4. Content-service infrastructural function: steps of file upload and share

smart space, she/he can request the file from the web server. The web server retrieves the file and sends it to the requester.

Since the Content-service publishes links on shared data on user's request and does not track smart space information changes it has no need to use the subscription operation and persistent connection to the SIB. Network connections to the SIB are established only for the time of request processing.

The proposed solution increases the application dependability due to delegating of large volumes data storing to a specialized storage. The smart space content is not overloaded by storing volumetric factual data. Support for keeping the data integrity is due to the use of verified file sharing web technologies.

V. Mechanisms for Fault Tolerance of Subscription

Let us consider mechanisms that support the fault tolerance of the subscription operation in a Smart-M3 application. The mechanisms are implemented on the side of infrastructural KPs. The enhanced infrastructure is shown in Fig. 5.

Subscription control mechanism: infrastructural KPs make active regular checks for subscribed data (e.g., on timer). It is used to determine failures in the subscription operation. There are two types of failures.

1) Notification has not been received although the subscription connection is still working.
2) Subscription connection with the SIB has been lost.

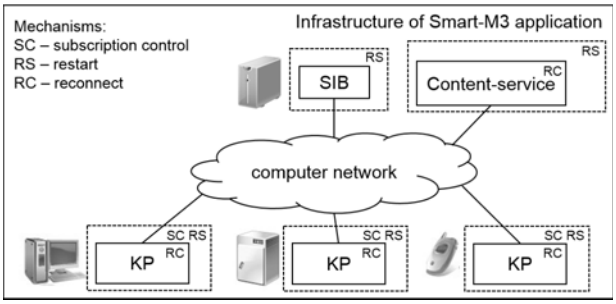

Fig. 5. Enhanced infrastructure for a Smart-M3 application

Checks for subscription can be implemented either in KP code or in KPI[2] middleware. The latter option makes the application developer free from programming own fault tolerance support since the control is performed automatically by the KPI. Note that Content-service does not need this subscription control mechanism because the service uses no subscriptions.

To implement an active check of subscription one can use the notification model introduced in [21]. In brief, a KP subscribes on its own notification type (a special set of RDF triples kept in the smart space). If such a notification is published in the smart space and the subscription has not delivered it to the KP then the infrastructural KP can check presence of unprocessed notification triples by itself and continue processing. Checks can be performed either automatically or manually.

If a failure with subscription has been detected, the following mechanism can be used for recovering. For simplicity we assume that each service is implemented with one infrastructural KP.

Restart/reconnection: an infrastructural KP reestablishes its network connection with the smart space (i.e., with SIB) and restores the previous state. Since the network connection is reestablished the subscription is also reset. The key point is restoration of the previous service state (the state the service was in before the failure). Service restart/reconnection is performed automatically after the failure detection. Manual intervention of a Smart-M3 application administrator is also allowed.

Reconnection implies that the service tries to reestablish the KP connection with the SIB during runtime, so the service process remains working. Therefore, connection reestablishing is performed by means of the service itself. On the other hand, during restart the service process is terminated, the KPs implementing this service are started again and connection is established once more. Restart is performed by means of the program environment where the service is launched, i.e., by means of operating system. Restart is convenient for use with console-oriented KPs (they also may run on a remote server as daemons). A console application or daemon can be quickly shut down and started again. In contrast, reconnection is essential for GUI-oriented KPs because they cannot be closed and then started again without affecting the users (at least, it cannot be hidden from the users). Although reconnection may be used in console-oriented KPs but it requires adding signal handlers into the KP code, thus making their development more complicated.

[2]Knowledge Processor Interface. It provides operations on accessing and interacting with smart space

TABLE III. USE OF MECHANISMS FOR SUBSCRIPTION FAULT TOLERANCE IN SMARTROOM

Infrastructure element	Restart	Reconnection
SIB	+ (auto)	−
Conference-service	+ (auto)	−
Agenda-service	+ (manual)	+ (manual)
Presentation-service	+ (manual)	+ (manual)
Content-service	+ (manual)	−

The proposed mechanisms of subscription control and restart/reconnection allow to increase the application dependability. They support detection of failures during subscription, which is treated as the most sensitive Smart-M3 operation, and subsequent recovery from such failures.

Consider the use of these mechanisms in the SmartRoom system (Table III). The SIB and Conference-service are console applications. Content-service is a web application. They are launched on the same machine since it is necessary to make this components working continuously and to have stable access to the smart space. The server machine is working under Ubuntu Linux OS, which provides an effective tool for launching console applications as services (daemons): Upstart event-based init daemon.[3] It controls services starting and stopping. Furthermore, it provides automatic service respawning: if there was failure in an infrastructural KP and it crashed then the KP will be restarted. Upstart also supports starting and stopping services in a chain: after starting a service, dependent services are started automatically. Therefore the SIB and Conference-service are launched as daemons on the server machine and Upstart performs their reconnection.

Launching the SIB (which consists of two applications: redsibd and sib-tcp) and Conference-service on the server machine is performed in the specific order:

$$\text{redsibd} \rightarrow \text{sib-tcp} \rightarrow \text{conference-service}$$

For example, see Upstart configuration file for Conference-service below.

```
# launch service after sib-tcp
start on started sib-tcp
# stop service before sib-tcp stopping
stop on stopping sib-tcp
# restart service on crash
respawn
# run service
exec /usr/bin/conference-service
```

This Upstart configuration file specifies conditions when the service is started and stopped and what dependent services to run. Notable configuration statement is `respawn`. It specifies restarting the service whenever it crashes.

The same way, the processes of redsibd and sib-tcp are launched. Table IV shows the dependencies described in configuration files for the SIB and Conference-service.

[3]http://upstart.ubuntu.com

TABLE IV. DEPENDENCIES FOR LAUNCHING THE SIB AND CONFERENCE-SERVICE IN SMARTROOM

Process	Launching dependencies
redsibd	start on startup (launches with OS)
sib-tcp	start on started redsibd stop on stopping redsibd
conference-service	start on started sib-tcp stop on stopping sib-tcp

In such manner, restart of console services is organized in the SmartRoom system. After launching redisbd, the other processes are started in a chain. If one of them is restarted (restart can be produced automatically by `respawn`) the dependent ones will be also restarted.

Since Content-service is a web application, it does not need a persistent network connection with the SIB. Content-service launching is controlled by Apache web server, and connection with the SIB is established only during processing of a user request. If a failure occurs then restarting Content-service needs restarting the web server.

In GUI-oriented services, such as Agenda-service and Presentation-service, restart is performed manually. (One has to close the application and start it again.) This way is not appropriate in SmartRoom since the services continuously display information: activity agenda and speaker's presentation. Their closing will spoil the visual experience of participants. The preferable way here is reconnection, which can be performed manually by an administrator. The subscription is recovered without termination of the service operation.

VI. EVALUATION

The considered above Content-service and mechanisms for subscription fault tolerance all support the dependability of Smart-M3 applications. They improve such attributes as reliability, integrity and maintainability. This section evaluates the proposed solutions. We analyze reduction of negative consequences for possible failures and confirm that the solutions are acceptable for such systems like SmartRoom.

A. Content-service

Let files be stored directly in the smart space. If failures occur (with SIB, network or sending KP) during file saving or processing then such a file will be corrupted or lost. Smart space content is represented as a set of RDF triples. Converting volumetric factual data to triples is complex and time-consuming task. If a failure interrupts such conversion then the data integrity is broken and the operation should be started from the beginning. Content-service uses matured web technologies for file receiving and reliable file system for storing. It clearly improves the application reliability and data integrity.

Another failure occurs if a Smart-M3 application processes a file stored on a remote host. If there are network problems or the host is shut down the file becomes unreachable. Content-service duplicates the file locally in advance. Thus, it ensures that the file will be available.

Content-service automates (to some extend) the process of file collecting. In such systems as SmartRoom the participants should provide their presentational content to organizers. In manual file collection, faults frequently occur due to the human factor. The user can easily make an error when collecting or saving files. Content-service defines a uniform way to collect files, thus reducing the human factor.

Although Content-service enhances separate attributes of the application dependability (reliability, integrity), it still acts as a centralized solution, with certain limits of the dependability. Nevertheless, a distributed storage can easily make the system complicated and worsen the dependability due to the increased number of elements. Our deployments of SmartRoom system in various practical setting confirms that such a centralized solution is acceptable for the considered class of Smart-M3 applications. Note that many failures in Content-service can be controlled and repaired by other mechanisms.

B. Mechanisms for subscription fault tolerance

Subscription control is intended to determine a failure in two situations: 1) when subscription indication was lost and 2) when subscription connection was disrupted. In the first situation, the mechanism checks for subscribed data and in case of changes it means that a failure occurred. Hence proactive subscription operation is augmented with active checking on time. In the second situation, connection disruption is determined and subscription is restored.

In these mechanisms, the application reliability and maintainability are improved due to control that the subscription is working and notifications on changed data are received. If a failure has occurred then the application is recovered and continues its operation. Nevertheless, it is not always possible to detect correctly information changes. (Between subsequent checks, some data are changed and then the data are changed back.) Also there is no clear criterion on how to identify breaks in a subscription connection.

Recently, this problem results in development of mathematical methods to estimate the frequency of such checks. For the Smart-M3 case, a model of subscription check interval is considered in [22]. Note, however, that approach is focused on client KPs. In contrast to infrastructural KPs, client KPs are mobile and their individual operation is less crucial for the whole application operation.

Restart/reconnect is a subsequent action of the subscription control. The action recovers application after a detected failure. It is effective against failures in operation of a particular infrastructural element: freezes, crashes, or misbehavior. Apart from the subscription control, additional control mechanisms are still needed to detect other types of failures and to invoke restart/reconnection.

C. Performance

We measured the time that is needed to restart and reconnect the SmartRoom services. The experimental setup includes a server computer (Ubuntu Linux, Intel Xeon 2.30GHz, 4GB RAM) for hosting the SIB, Conference-service, and Content-service. The server is located in the corporate network, not in the same LAN of SmartRoom.

A personal computer (Windows, Intel Core 2 Quad 2.40GHz, 8GB RAM) is used to run Agenda-service and Presentation-service. The computer is in the same LAN with

TABLE V. SERVICE AVAILABILITY: RESTORATION TIME

Infrastructure element	Restart (sec)	Reconnection (sec)
SIB	1.045	–
Conference-service	2.023	–
Agenda-service	1.350	0.134
Presentation-service	0.230	0.075
Content-service	2.193	–

mobile SmartRoom clients. In the case of restart, the time between service shut down and full start up (readiness of service to process requests) was measured. In case of reconnection, the time from the disconnect point until service readiness was measured. For each experiment, 10 samples were measured. The experimental results are summarized in Table V.

We may conclude that both restart and reconnection mechanisms are rather fast. The time is less than 2 seconds, which is appropriate for the considered class of applications.

For evaluation of the Content-service efficiency the file upload time is estimated. Upload time was measured on two popular web servers: Apache and nginx. They represent the "heavy" and "lightweight" server types, respectively. Testing on different servers analyzes the dependency of Content-service on hosting middleware. To upload a 10Mb-file, it takes 1.814 seconds on average (standard deviation is 0.014) and 1.842 seconds (standard deviation is 0.030) for Apache and nginx, respectively. The size of 10Mb shows the worst case since it is the maximal permitted size for presentation in the SmartRoom system. Therefore, Content-service runs with similar efficiency on both considered web servers and takes less than 2 seconds to upload a presentation.

VII. CONCLUSION

This paper particularized the notion of software infrastructure for the case of Smart-M3 applications. We presented two solutions for the fault tolerance; both are implemented on the software infrastructure level of a given application. The first solution adds to the infrastructure a dedicated service to control of content. This Content-service improves data safety and integrity when the application operates with volumetric factual data. The second solution consists of mechanisms for subscription control (fault detection) and recovery (when fault has happened). Using the SmartRoom system as a use case study, we show the applicability of our solutions for the real-life Smart-M3 application. We experimentally confirm the efficiency of the proposed solutions.

ACKNOWLEDGMENT

This research is financially supported by project # 1481 from the basic part of state research assignment # 2014/154 of the Ministry of Education and Science of the Russian Federation and by research project # 14-07-00252 of the Russian Fund for Basic Research. The work is a part of project 14.574.21.0060 (RFMEFI57414X0060) of Federal Target Program "Research and development on priority directions of scientific-technological complex of Russia for 2014–2020". The article was published with financial support from the Strategic Development Program of Petrozavodsk State University. The authors are grateful to the Open Innovations Association FRUCT for its support and R&D infrastructure. We would also like to thank anonymous reviewers for their valuable comments on this paper.

REFERENCES

[1] D. J. Cook and S. K. Das, "How smart are our environments? an updated look at the state of the art," *Pervasive and Mobile Computing*, vol. 3, no. 2, pp. 53–73, 2007.

[2] H. Chen, T. Finin, A. Joshi, L. Kagal, F. Perich, and D. Chakraborty, "Intelligent agents meet the semantic web in smart spaces," *IEEE Internet Computing*, vol. 8, pp. 69–79, November 2004.

[3] S. Balandin and H. Waris, "Key properties in the development of smart spaces," in *Proc. 5th Int'l Conf. Universal Access in Human-Computer Interaction (UAHCI '09). Part II: Intelligent and Ubiquitous Interaction Environments, LNCS 5615*, C. Stephanidis, Ed. Springer-Verlag, 2009, pp. 3–12.

[4] D. Korzun, S. Balandin, and A. Gurtov, "Deployment of Smart Spaces in Internet of Things: Overview of the design challenges," in *Proc. 13th Int'l Conf. Next Generation Wired/Wireless Networking and 6th Conf. on Internet of Things and Smart Spaces (NEW2AN/ruSMART 2013), LNCS 8121*, S. Balandin, S. Andreev, and Y. Koucheryavy, Eds. Springer-Verlag, Aug. 2013, pp. 48–59.

[5] J. Honkola, H. Laine, R. Brown, and O. Tyrkkö, "Smart-M3 information sharing platform," in *Proc. IEEE Symp. Computers and Communications (ISCC'10)*. IEEE Computer Society, Jun. 2010, pp. 1041–1046.

[6] D. D. Corkill, "Collaborating Software: Blackboard and Multi-Agent Systems & the Future," in *Proc. the Int'l Lisp Conference*, October 2003, invited paper.

[7] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, pp. 114–131, June 2003.

[8] A. Smirnov, A. Kashnevik, N. Shilov, I. Oliver, S. Balandin, and S. Boldyrev, "Anonymous agent coordination in smart spaces: State-of-the-art," in *Proc. 9th Int'l Conf. Next Generation Wired/Wireless Networking (NEW2AN'09) and 2nd Conf. Smart Spaces (ruSMART'09), LNCS 5764*. Springer-Verlag, 2009, pp. 42–51.

[9] A. A. Lomov and D. G. Korzun, "Subscription operation in Smart-M3," in *Proc. 10th Conf. of Open Innovations Association FRUCT and 2nd Finnish–Russian Mobile Linux Summit*, S. Balandin and A. Ovchinnikov, Eds. SUAI, Nov. 2011, pp. 83–94.

[10] W. Xie, Y. Shi, G. Xu, and Y. Mao, "Smart platform — a software infrastructure for smart space (SISS)," in *Proc. 4th IEEE Int'l Conf. on Multimodal Interfaces (ICMI '02)*. IEEE Computer Society, 2002, pp. 429–434.

[11] X. Wang, J. S. Dong, C. Chin, S. R. Hettiarachchi, and D. Zhang, "Semantic space: An infrastructure for smart spaces," *Computing*, vol. 1, no. 2, pp. 67–74, 2002.

[12] S. Sathish and C. di Flora, "Supporting smart space infrastructures: a dynamic context-model composition framework," in *Proc. 3rd Int'l Conf. on Mobile Multimedia Communications*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, p. 67.

[13] C. A. da Costa, A. C. Yamin, and C. F. R. Geyer, "Toward a general software infrastructure for ubiquitous computing," *IEEE Pervasive Computing*, vol. 7, no. 1, pp. 64–73, 2008.

[14] I. Galov and D. Korzun, "The smartroom infrastructure: Service runtime reliability," in *Proc. 14th Conf. of Open Innovations Association FRUCT*, S. Balandin and U. Trifonova, Eds. SUAI, Nov. 2013, pp. 188–189.

[15] D. Korzun, I. Galov, and S. Balandin, "Development of smart room services on top of smart-m3," in *Proc. 14th Conf. of Open Innovations Association FRUCT*, S. Balandin and U. Trifonova, Eds. SUAI, Nov. 2013, pp. 37–44.

[16] D. Korzun, I. Galov, A. Kashevnik, and S. Balandin, "Virtual shared workspace for smart spaces and M3-based case study," in *Proc. 15th Conf. of Open Innovations Association FRUCT*, S. Balandin and U. Trifonova, Eds. ITMO Univeristy, Apr. 2014, pp. 60–68.

[17] A. Vasilev, I. Paramonov, S. Balandin, E. Dashkova, and Y. Koucheryavy, "Mechanism for context-aware substitution of Smart-M3 agents

based on dataflow network model," in *Proc. Int'l Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE, 2013, pp. 113–117.

[18] R. Kadirov, E. Cvetkov, and D. Korzun, "Sensors in a Smart Room: Preliminary study," in *Proc. 12th Conf. of Open Innovations Association FRUCT and Seminar on e-Tourism*, S. Balandin and A. Ovchinnikov, Eds. SUAI, Nov. 2012, pp. 37–42.

[19] D. G. Korzun, I. V. Galov, A. M. Kashevnik, N. G. Shilov, K. Krinkin, and Y. Korolev, "Integration of Smart-M3 applications: Blogging in smart conference," in *Proc. 4th Conf. Smart Spaces (ruSMART'11) and 11th Int'l Conf. Next Generation Wired/Wireless Networking (NEW2AN'11)*. Springer-Verlag, Aug. 2011, pp. 51–62.

[20] M. R. Lyu *et al.*, *Handbook of software reliability engineering*. IEEE computer society press CA, 1996, vol. 222.

[21] I. Galov and D. Korzun, "Notification model for Smart-M3 applications," in *Proc. 14th Int'l Conf. Next Generation Wired/Wireless Networking and 7th Conf. on Internet of Things and Smart Spaces (NEW2AN/ruSMART 2014), LNCS 8638*, S. Balandin, S. Andreev, and Y. Koucheryavy, Eds. Springer-Verlag, Aug. 2014, pp. 121–132.

[22] A. Vdovenko and D. Korzun, "Active control by a mobile client of subscription notifications in smart space," in *These Proceedings*, Oct. 2014.