Project no.	034084
Project acronym:	SELFMAN
Project title:	*Self Management for Large-Scale Distributed Systems based on Structured Overlay Networks and Components*

European Sixth Framework Programme
Priority 2, Information Society Technologies
Publishable Final Activity Report
Year Four (M40)

Due date of deliverable:	Nov. 15, 2009
Actual submission date:	Nov. 30, 2009
Start date of project:	June 1, 2006
Duration:	40 months
Dissemination level:	CO

Contents

1 The Challenge: Large Internet Applications

As Internet applications become larger and more complex, the task of managing them becomes overwhelming. "Abnormal" events such as software updates, faults, threats, and performance hotspots become normal and even frequent occurrences. The goal of SELFMAN is to handle these events automatically by making the applications self managing. They will reconfigure themselves to handle changes in their environment or requirements without human intervention but according to high-level management policies. We focus on four axes of self management, namely self configuration, self healing, self tuning, and self protection.

The SELFMAN project[1] approached the problem by combining the complementary strengths of structured overlay networks and advanced component models. Structured overlay networks are already self-organizing. They originated with peer-to-peer file sharing applications, but have matured to provide strong guarantees and efficient communication and storage operations. As such, they form an ideal foundation for hosting self-managing services. To achieve this, we leverage the overlay network by rebuilding it using advanced component models. These make the overlay network modular and add the hooks needed for self management. The hooks provide introspection, reflections, and dynamic reconfiguration abilities. In this way, the overlay network can host an application that manages itself.

SELFMAN has reconstructed the structured overlay network as part of a self-managing component architecture and used it to build high-level self-managing services. We have built powerful services including transactional replicated storage and media streaming and we have built scalable Internet applications on top of these services: a media streaming product, PeerTV, that provides competitive quality of service at much lower cost than existing products, a Distributed Wikipedia that is competitive in performance with the standard Wikipedia but is scalable, and a decentralized drawing application, DeTransDraw, for mobile phones (gPhones) that allows local editing while keeping global drawing coherence.

2 Major Project Results

2.1 Breakthrough results

SELFMAN has developed two breakthrough results as well as numerous scientific advances at all levels of self management for distributed systems. The

[1] ICT 6FP project, contract 034084, June 2006-Sep. 2009, `www.ist-selfman.org`.

breakthrough results are in distributed transactional storage and media distribution:

- **PeerTV**. PeerTV is a state-of-the-art application developed as a product by partner Peerialism that distributes video over Internet with live streaming and progressive download. It uses advanced technology based on peer-to-peer structured overlay networks, optimization algorithms, and component models. It uses the Chandelier dynamic peer-to-peer optimization algorithm and was developed with an integrated simulation tool MyP2PWorld, which radically sped up development and shorten time to market. It uses new techniques for firewall hole punching and NAT traversal of video streams. PeerTV has comparable QoS to leading distribution providers but at much lower costs. PeerTV was developed for and is being used by the Swedish company MPS Broadband AB.

- **Scalaris**. Scalaris is an open-source library that provides a scalable global storage service. Scalaris is built on top of a structured peer-to-peer network and implements a key/value store that supports transactions. It survives node crashes and network problems using replication and a sophisticated consensus algorithm. It scales to hundreds of nodes and provides strong data consistency in the face of concurrent operations, node failures, and network problems. It does load balancing with an algorithm that uses estimates of global knowledge to reduce the storage load deviation between nodes. Scalaris was used to implement a version of Wikipedia that won first prize in the IEEE International Scalable Computing Challenge 2008. It is competitive in performance to the actual Wikipedia backend (14,000 read+write transactions / second on a synthetic benchmark) but is more robust and scalable.

2.2 Major software results

SELFMAN has produced many other major and minor software results. We have selected the most important software:

- **Beernet**. Beernet is a modification of Scalaris that is based on a "relaxed ring" overlay network, which is easier to manage. It relaxes the connectivity condition of the ring underlying the structured overlay network. It requires only that a node be in the same ring as its successor (instead of both its successor and predecessor). Beernet also modifies the transaction algorithm to request locks quickly and to notify all

nodes of modified state. We have used Beernet to implement the collaborative drawing application DeTransDraw on gPhones: HTC Magic mobile phones running the Android operating system. DeTransDraw uses transactions to maintain global coherence of the drawing as well as to overcome network latency (edits are immediately shown locally while the transaction algorithm attempts to obtain a global commit).

- **Kompics**. Kompics is an advanced component model for building reconfigurable distributed systems from event-driven components. Kompics systems can be uniformly evaluated in large-scale reproducible simulation and distributed deployment, using both the same system code and the same experiment scenarios. Kompics components are concurrent and readily exploit multi-core architectures. They are decoupled by publish/subscribe ports and channels and are compositional. They can form dynamically reconfigurable architectures and fault supervision hierarchies.

- **CompOz**. CompOz is a complete self-configuration framework consisting of three libraries that work together: FructOz for dynamic deployment and configuration, LactOz for dynamic navigation and monitoring, and WorkflOz for dynamic distributed workflows. FructOz implements the Fractal component model on top of Oz. LactOz provides navigation and monitoring abilities on top of FructOz structures. WorkflOz allows the construction of workflows as FructOz structures, which allows LactOz to monitor workflow execution.

- **Mozart Programming System 1.4.0**. The Mozart system is a software development platform based on the Oz multiparadigm programming language. It supports highly dynamic applications and fine-grain concurrency. Mozart 1.4.0 provides an advanced transparent distribution subsystem that allows to develop distributed applications very similar to centralized applications. All language entities (objects, components, dataflow variables, threads, ports, etc.) have distribution protocols that let them work in a distributed setting. The Mozart 1.4.0 fault model allows fault-tolerance abstractions to be built within the language using its built-in asynchronous failure detection, thus keeping transparency. Mozart 1.4.0 was used to build the Beernet and CompOz libraries.

2.3 Major scientific results

SELFMAN has produced many scientific results. Here are the three most important:

- **Atomic transactions on a peer-to-peer network**. We have developed a transaction manager that works for data stored on structured peer-to-peer networks. The heart of the transaction manager is the algorithm that implements the atomic commit. It is an optimized version of Lamport's Paxos uniform consensus algorithm that needs only four communication steps instead of six in the common case when there are no failures. The improvement was possible by exploiting information from the data replication in the structured peer-to-peer network. The transaction manager works correctly under realistic Internet conditions, where at any time nodes can crash or the network can have problems. The Scalaris library uses the atomic commit algorithm to implement strong data consistency and atomic transactions.

- **Merge algorithm for network partitioning**. We have solved the network partitioning problem for structured overlay networks. If the network is partitioned, the overlay splits into several smaller overlays, which each continue to provide its service as best it can with the nodes it contains. If the partition goes away (the network is repaired), then the merge algorithm will automatically combine the smaller overlays back into a single large overlay, thus restoring the complete service. This behavior can be seen a reversible phase transition, in analogy with thermodynamics. It can be used to build extremely robust Internet applications that survive network partitioning.

- **Methodology for building self-managing applications**. The SELFMAN project has pushed the state of the art in software development techniques for self-managing applications. Out of the practical experience of SELFMAN, we have derived a development methodology for large-scale distributed systems based on the concept of *weakly interacting feedback structures*. A feedback structure is a hierarchy of interacting feedback loops that together maintain one global system property. This gives a much more natural and powerful way of designing large systems than the traditional layered approach. We have applied this methodology to decentralized distributed systems and in particular to the Beernet and Scalaris systems. Considered in this way, Scalaris consists of six weakly interacting feedback structures. We are continuing to develop and extend this methodology, especially for the development

of extremely robust applications, by generalizing the idea of reversible phase transitions.

3 Quantitative Project Results

We give some quantitative indicators of the results of SELFMAN, including publications, awards, workshops organized, advertisements and articles, and coordination events.

3.1 Publications and awards

During the whole project, SELFMAN made the following publications to disseminate its scientific results and software results:

- 7 papers in international journals
- 4 Ph.D. dissertations completed + 1 dissertation nearing completion.
- 30 papers in international conferences
- 6 book chapters
- 30 invited talks
- 37 workshop papers and technical reports

Three of these results won awards:

- Thorsten Schütt, Monika Moser, Stefan Plantikow, Florian Schintke, Tallat Shafaat, Seif Haridi, and Alexander Reinefeld. "A Transactional Scalable Distributed Data Store: Wikipedia on a DHT". *First Prize with plaque and monetary award.* 1st IEEE International Scalable Computing Challenge, Lyon, France, May 2008 (see Section 4 for more information).
- Artur Andrzejak and Luis Silva. "Using Machine Learning for Non-Intrusive Modeling and Prediction of Software Aging". *Best Paper Award.* 11th IEEE/ IFIP Network Operations and Management Symposium (NOMS 2008), Salvador de Bahia, Brazil, April 2008.
- Boris Mejias. "Beernet: A Relaxed-Ring Approach for Peer-to-Peer Networks with Transactional Replicated DHT". *Best Presentation Award* (of 13 submissions). Doctoral Symposium, XtreemOS Summer School, Sep. 2009.

3.2 Workshops

SELFMAN organized two workshops, both colocated with the SASO (the IEEE International Conference on Self-Adaptive and Self-Organizing Systems).

- "Decentralized Self Management for Grids, P2P, and User Communities". Colocated with SASO 2008, Venice, Italy, Oct. 20-24, 2008. Joint workshop with GRID4ALL project. 15 papers accepted.
- "Architectures and Languages for Self-Managing Distributed Systems". Colocated with SASO 2009, San Francisco, CA, Sep. 14-18, 2009. Invited speaker: Terence Kelly, HP Labs, "Discrete Control Theory for Self-Managing Systems". 8 papers accepted.

3.3 Advertisements and articles

The following table gives a complete list of all advertisements and advertorials placed by the SELFMAN project and also lists as far as we know the major articles related to SELFMAN that have appeared in the technical press.

Date	Type	Type of audience	Countries addressed	Size of audience	Partner
Jun. 2006	Project website	researchers	world		UCL
Nov. 2006	0.5-page ad Parliament Magazine	deciders	European Parliament	thousands	UCL
Aug. 2007	Almende Summer School	researchers	Europe	appx. 50	UCL
May 2008	IEEE Competition first prize	researchers	world		ZIB
Jun. 2008	3-page article eStrategies Projects	deciders	Europe	39,000	UCL
Oct. 2008	Workshop SASO 2008	researchers	world	appx. 50	UCL
Nov. 2008	2-page article Research Review	deciders	Europe	thousands	UCL
Nov. 2008	3-page article TheServerSide	IT industry	world		ZIB
Dec. 2008	Web banner PSCA Int.	deciders	Europe		UCL
Aug. 2009	Press on GGF acquisition	IT industry	world		PEER
Sep. 2009	Workshop SASO 2009	researchers	world	appx. 50	INRIA
Sep. 2009	Web banner PSCA Int.	deciders	Europe		UCL
Oct. 2009	2 articles ICT Results	researchers	world		UCL
Oct. 2009	1 article ACM Technews	researchers	world		UCL
Oct. 2009	>15 blogs	IT industry	world		UCL
Oct. 2009	1-page article Automatisering Gids (Dutch)	IT industry	Netherlands	thousands	UCL
Jan. 2010	1-page article PS Review	deciders	Europe	thousands	UCL
Jan. 2010	1-page editorial PS Review	deciders	Europe	thousands	UCL

3.4 Coordination events

SELFMAN participated in a number of European coordination events.

- Collaboration through CoreGRID Network of Excellence (four SELFMAN partners were members).
- FMCO 2007 (Oct. 2007) (Formal methods, components, and object technology): five projects presented (ARTIST, SELFMAN, SENSORIA, MODELWARE, and CREDO). SELFMAN gave four talks.
- Internet of Services Collaboration Meeting for FP6 and FP7 (Sep. 2008).
- European Future Technologies Conference (FET09), Apr. 2009.
- Panel on Future Internet Services at the Future of Internet Conference (FIA09), May 2009.
- DeTransDraw demonstrator at Internet of Services 2009, June 2009.

4 Pilots

The SELFMAN project developed two major pilot systems, Scalaris and PeerTV. We summarize both systems below.

4.1 Scalaris

Scalaris is a self-managing scalable, transactional database that is suitable for building global services. It provides a distributed key/value store on the top of a replicated storage layer and an enhanced structured P2P overlay network. The service is highly available and supports strong data consistency in the face of concurrent data operations, computer failures, and network problems. Its implementation uses a fast consensus protocol with low communication overhead that has been optimally embedded into the overlay network. Scalaris and similar systems will be core components in future Cloud Computing environments and Web 2.0 services. Scalaris is designed and implemented by the Zuse Institute Berlin (ZIB) in collaboration with KTH/SICS in the SELFMAN project.

We demonstrated the capabilities of Scalaris by reimplementing the core of Wikipedia (see Figure 1). This implementation won first prize in the IEEE International Scalable Computing Challenge 2008 (the first-prize plaque is

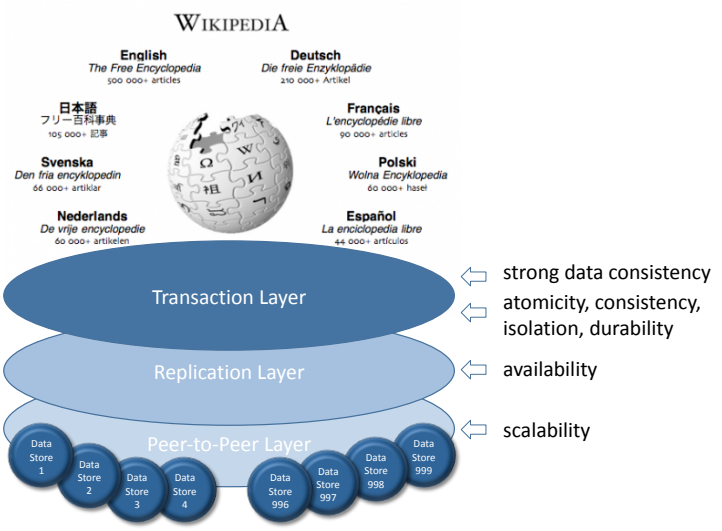

Figure 1: Implementing Wikipedia with the Scalaris self-managing transactional database

shown in Figure 2). The implementation is both fast and scalable: using eight servers it executes 2,500 transactions per second. All operations are performed within transactions to guarantee data consistency and replica synchronization. Adding more computers improves the performance almost linearly. The public Wikipedia, in contrast, employs ten servers to execute the 2,000 requests per second that hit the backend on its large master/slave MySQL database (Wikipedia handles around 50,000 requests per second, with 48,000 handled by proxies).

For many Web 2.0 services, the total cost-of-ownership is dominated by the costs needed for personnel to maintain and optimize the service. Scalaris greatly reduces these costs with its built-in self-management properties. It is self healing: when it detects a computing node crash or network problem, it immediately repairs its P2P overlay network and the database. Management tasks such as adding or removing nodes require no or very little human intervention. It is self tuning: it autonomously moves items to distribute the load evenly over the system to improve the response time. When deploying Scalaris over multiple data centers, these algorithms are used to place frequently accessed items near the users.

4.2 PeerTV

PeerTV is a comprehensive application for end users, content owners, and broadband operators that distributes video over Internet with live streaming

Figure 2: Scalaris won first prize in the 2008 IEEE International Scalable Computing Challenge.

and progressive download (see Figure 3). It was launched in late 2008 in the Swedish market by startup company Peerialism. It offers its customers lower cost of distribution, reduced peak capacity investments, as well as higher viewing capacity (more simultaneous viewers), through advanced technology such as traffic optimization and P2P overlay networks.

To reach the objectives for PeerTV and to meet other customer requirements, Peerialism has developed advanced technology in several areas: innovative P2P components including P2P protocols and optimization algorithms (such as Chandelier) targeted towards live streaming, an integrated simulation and emulation tool (MyP2PWorld) to radically speed up development and shorten time to market, and new techniques for firewall hole punching and NAT traversal of video streams. The result is a new, highly capable video

Figure 3: Media streaming with the PeerTV product

distribution solution built on structured P2P overlays developed from scratch in little over 12 months. PeerTV has comparable QoS to leading distribution providers but at much lower costs. PeerTV is simple to deploy and friendly to the ISP. The optimization engine takes into account the network topology, minimizing the traffic load in the network. Deployment is done without any changes to either the streaming server or the media client. PeerTV acts as a transparent proxy—any video format is wrapped in a PeerTV transport envelope—making the media player believe that it is still talking directly to the server using standard RTP/RTSP protocols.

A key success factor in the current and future development work is direct access to advanced research in distributed systems. Peerialism has close ties to the Swedish Institute for Computer Science (SICS) and the Royal Institute of Technology (KTH) and is a partner in several Swedish and European projects, in particular the SELFMAN project. The connections with SELFMAN and these institutes allow the company to quickly test and evaluate

new P2P components. The company receives valuable feedback on its own work and is provided access to work by others; for instance, the company is currently testing the Kompics component model Kompics developed in SELFMAN and is investigating potential benefits of closer integration with the CLIF load injection framework developed by SELFMAN partner France Telecom, and the scalable transactional store Scalaris developed in SELFMAN by ZIB and KTH.

Peerialism currently employs 14 people with offices in Stockholm and Cairo. It finances its commercialization and growth out of its own revenues.

5 Availability of Results

The following SELFMAN software results are available at the following URLs:

PeerTV media streaming product	`www.peerialism.com`
Scalaris transactional store	`scalaris.googlecode.com`
Beernet transactional store	`beernet.info.ucl.ac.be`
Kompics reactive component model	`kompics.sics.se`
Mozart Programming System	`www.mozart-oz.org`
SicSim discrete event simulator for P2P	`www.sics.se/~amir/sicsim`

The following table gives a more complete list of software results of SELFMAN and their potential for exploitation. Some of this software has not yet been released or is subject to restrictions in use.

Exploitable knowledge	Exploitable product(s)	Appl. sector	Date for comm. use	Patents or IPR	Owner
Transaction algorithm	Scalaris	Internet	2009	license	ZIB
Transaction algorithm	Beernet	Internet	2010	license	UCL
Media streaming	PeerTV	Internet	2009	trade secret	PEER
Simulation	MyP2PWorld	software		license	PEER
Self protection	Wikimedia Credibility Extension	Internet	2009	license	NUS
Simulation	SWN simulator	software	2010	license	NUS
Simulation	SicSim	software	2009	license	KTH
Component model	Kompics	software	2010	license	KTH
Development platform	Mozart 1.4.0	software	2008	license	UCL
Deployment & configuration library	FructOz (in CompOz)	software	2010	license	INRIA
Navigation & monitoring library	LactOz (in CompOz)	software	2010	license	INRIA
Workflow library	WorkflOz (in CompOz)	software	2010	license	INRIA

6 Potential Impact of Results

6.1 Scalaris and Beernet

Both Scalaris and Beernet are the subject of significant uptake.

- Scalaris was released under the Apache Open Source License. It is under continuing development by a community of developers (not just within SELFMAN). The ZIB spin-off company onScale is doing support and customization for Scalaris.
- Scalaris is being evaluated and used by many companies. We give a small sampling of the companies we know who are evaluating it for internal use:

  - Linden Lab, maker of Second Life application.
  - StudiVZ, a German social network, rival to Facebook.
  - Plurk, a social network, rival to Twitter.
  - searchmetrics.com, a search optimization company.
  - immobilienscout24.de, a German real estate agent.

- Beernet was released under the BSD Open Source License.
  - The DeTransDraw application and its gPhone implementation were done in collaboration with the Belgian Laboratory of Computer-Human Interaction (BCHI), headed by Jean Vanderdonckt.
  - The RELEASeD research group at UCL, headed by Kim Mens, is implementing the relaxed ring in Objective-C for use in Apple devices including the iPhone.
  - The SOFT laboratory at the Vrije Universiteit Brussel plans to use the relaxed ring and the feedback methodology for context-aware systems.

6.2 Peerialism acquisition

SELFMAN partner Peerialism was the subject of an acquisition attempt by software company Global Gaming Factory X (GGF). GGF (Global Gaming Factory X) would acquire the file sharing site The Pirate Bay (the eighth most popular site on the Internet) as well as Peerialism. Peerialism would have become the technology provider for a legal version of The Pirate Bay. The following text is from MarketWatch, June 30, 2009 (`www.marketwatch.com`):

> Following the completion of the acquisitions, GGF intends to launch new business models that allow compensation to the content providers and copyright owners. The responsibility for, and operation of the site will be taken over by GGF in connection with closing of the transaction, which is scheduled for August 2009.
>
> GGF has entered into an agreement to acquire the shares in Peerialism AB. Peerialism AB is a software technology company with its origin in KTH Royal Institute of Technology and SICS, Swedish Institute of Computer Science and which presently is owned by the employees. The owners as well as the employees will continue to work for the company. Peerialism develops solutions for data distribution and distributed storage based on new

> p2p-technology. The access to the technology is secured by the acquisition. The consideration amounts to in aggregate MSEK 100 [10 MEuro].
>
> "Peerialism has developed a new data distribution technology which now can be introduced on the best known file-sharing site, The Pirate Bay. Since the technology is compatible with the existing it will quickly allow for new values to be created for all key stakeholders and facilitate new business opportunities," says Johan Ljungberg, CEO Peerialism.
>
> "As a result of the acquisitions of The Pirate Bay and Peerialism, GGF will have a strategic position in the international digital distribution market. File sharing traffic is estimated to account for more than half of todays global Internet traffic. The Pirate Bay has a global brand and holds a key position with over 20 million visitors and over one billion page views per month," says Hans Pandeya, [CEO GGF].

The deal did not go through due to financial problems at GGF. Nevertheless, the final result is positive since the acquisition attempt has provided a global visibility for Peerialism and its unique technologies, which were developed partly in SELFMAN. Peerialism is ranked among the 33 hottest tech startups in Sweden in 2009 according to the Swedish technology newspaper NyTeknik[2] and one of the top three technology companies in Europe according to the Seedcamp community.[3] At the end of 2009, Peerialism is being praised internationally as a company that is promoting the use of research-based P2P technology in innovative ways for media distribution.

6.3 Media coverage

SELFMAN has received significant media coverage:

- *Peerialism acquisition.* The Swedish company GGF attempted in Aug. 2009 to acquire The Pirate Bay together with SELFMAN partner Peerialism. This is explained in the previous section.
- *Articles in the press.* SELFMAN has been the subject of numerous articles on the Internet since Oct. 2009, in online technology newspapers, blogs, and the paper newspaper Automatisering Gids (in Dutch).

[2]Here are Sweden's hottest technology startups (in Swedish), November 2009. Available at `www.nyteknik.se/nyheter/innovation/tillvaxtforetag/article543431.ece`.

[3]Mini Seedcamp Helsingborg winners, May 2009. Available at `blog.seedcamp.com/2009/05/mini-seedcamp-helsingborg-winners.html`.

Articles have appeared in ICT Results, ACM Technews, ScienceDaily, MegaPlatinum, VWN News, NUZE.ME. Newstin, Fast Company, ReadWriteWeb, PhysOrg.com, Technology.am, AlphaGalileo, and The Web Scene. Foreign language articles have appeared in French, Russian, and Portuguese. The technology blog RobotArmageddon has compared SELFMAN to Skynet (which takes over the world in the Terminator films) because of its self-defending and self-healing characteristics.

We give a selection of quotes from the articles about SELFMAN:

- "It will take the internet to the next level."
- "A clever algorithm called Paxos does the heavy lifting to make sure the nodes all agree."
- "PeerTV found that they could provide the same quality of service as established video providers, [...] but with a cost reduction of from 50 to 90 percent."
- "Projects like SELFMAN are frightening because more advanced versions with artificial intelligence will eventually become available."
- "When computers crash, Scalaris does not stop but at worst just slows down."
- "The system works best for Internet applications where people collaborate and the system has to maintain coherence."
- "[...] many applications can be made SELFMAN ready."

6.4 Collaboration with other projects

The SELFMAN project's results have had a significant influence on other projects. We give the most important ones:

- *XtreemOS project*: Part of Scalaris (the DHT and transaction framework) was used to implement a scalable metadata store and a publish/-subscribe system. As a result, the Mandriva Linux distribution is now offering Scalaris packages (RPMs).
- *GRID4ALL project*: This project developed a framework for self-managing component-based applications that extends the Fractal model. We also collaborated on an initial methodology on using feedback loops in system design. This work influenced the SELFMAN work on methodology for building self-managing applications.

- *MANCOOSI project*: We have a collaboration on distributed information sharing to improve the efficiency of package installability solving which is highly compute-intensive.
- *Almende Summer School* (Aug. 2007): We gave a tutorial on self-managing software. Almende is a company focused on innovative self-organizing and adaptive software for companies.

6.5 Follow-up project proposals

The SELFMAN project has led to a number of project proposals submitted during the project's final period (July-Sep. 2009), three in the FP7 Call 5 and one in ComplexityNet. Four of these proposals are direct successors to SELFMAN:

- *PHASEMANIA*: An FP7 STREP in FET Open on using reversible phase transitions to build extremely robust self-managing Internet applications. UCL, ZIB, SICS, and Peerialism are partners. The self-management methodology and the network partitioning solution (merge algorithm) of SELFMAN are key starting points.
- *CLOUDMAN*: An FP7 STREP in Objective 1.2 (Internet of Services) on a self-managing peer-to-peer cloud infrastructure. UCL, ZIB, SICS, Peerialism, and INRIA are partners. The Scalaris/Beernet technology developed in SELFMAN is a key part of the peer-to-peer cloud.
- *VISION*: An FP7 IP on a federated data intensive storage cloud. KTH is a partner in the area of computational storage. The Kompics technology developed in SELFMAN will be used and further developed.
- *3C (Conjuring Clouds with Constraints)*: An Expression of Interest in the European ComplexityNet initiative, on using constraints for dynamic scheduling of clouds. UCL is a partner in the area of self-managing distributed applications.

7 Lessons Learned During the Project

During the SELFMAN project we learned many lessons about how to build practical self-managing distributed applications and about the promising future directions of this work.

7.1 Future directions and the SELFMAN legacy

The SELFMAN project has pushed the state-of-the-art in Internet applications. From the viewpoint of the project, there are many directions in which SELFMAN's legacy can influence future research and application development:

- Cloud computing, and specifically, *peer-to-peer cloud computing*. Cloud computing is the practical realization of computing as a utility. Current clouds are implemented in datacenters. A peer-to-peer cloud is complementary to this. It implements the cloud on a peer-to-peer overlay network and uses advanced techniques to achieve elasticity and high availability.

- *Practical methodology for self-managing Internet applications*. This is based on designing with interacting feedback structures, which is a practical way to achieve separation of concerns in a self-managing context. We will also design for a desired global behavior using the phase space, which defines the behavior of an application in predictable fashion depending on macroscopic variables that characterize the operating conditions of the application.

- *Next generation of Internet applications*. The SELFMAN results are directly applicable to the following application areas:

  - Applications for mobile devices (gPhone, iPhone, tablet).
  - Future collaborative intelligence (combining knowledge of a community of users to increase overall intelligence). This is important for the next generation of recommendation systems and for open source communities.
  - Future social networks and collaborative tools (the next generation of Wikipedia, Facebook, and Twitter).

7.2 How to design self-managing applications

As Internet programs become larger and more complex, designing them and predicting their behavior become daunting. In addition to users coming and going and acting concurrently, "abnormal" events such as software errors, partial failures, attacks, and hotspots become normal. To address this problem, we have learned that that these programs should be designed from the start as a set of *interacting feedback structures*. Each feedback structure consists of one or more feedback loops and continuously *maintains one global*

system property. In a well-designed system, *no part should exist outside of a feedback structure*. We have come to this conclusion from studies of existing robust systems taken from biology and computing.

The Scalaris transactional store shows the power of this approach. It combines a structured peer-to-peer network, a replicated storage layer, and a transaction layer. As a complement to the usual layered view of such systems (see for example Figure 1), we can explain it as *six feedback structures* working together in a harmonious way. Scalaris scales smoothly and efficiently to hundreds of nodes, handles node and network failures, and performs load balancing. Scalaris uses a modified Paxos uniform consensus algorithm to implement atomic commit.

In our methodology, a system's specification consists of a conjunction of properties, each of which is implemented by one feedback structure. This *achieves separation of concerns* by defining the concerns in terms of the feedback structures that naturally implement them. We are continuing our study of how to design with this approach. We are extending the approach to design for a desired global behavior using *reversible phase transitions*. Such systems will be much easier to design, predict, and manage, and will be less subject to global problems such as multicast storms, chaotic behavior, and cascading failures. They will provide well-defined behavior for a wide range of environmental conditions, even extremely hostile ones.

7.3 How to solve the network partitioning problem

We have solved the network partitioning problem for structured peer-to-peer networks. Any long-lived Internet-scale distributed system is destined to face network partitions. Although the problem of network partitions and mergers is highly related to fault tolerance and self management in large-scale systems, it has hardly been studied in the context of structured peer-to-peer systems. These systems have mainly been studied under churn (frequent node joins, leaves, and failures), which as a side effect solves the problem of network partitions, as it is similar to massive node failures. Yet, the crucial aspect of network mergers has been ignored. In fact, it has been claimed that ring-based structured overlay networks, which constitute the majority of the structured overlays, are intrinsically ill-suited for merging rings. We have designed and built a practical algorithm for merging multiple similar ring-based overlays when the underlying network partition goes away. We examine the solution in dynamic conditions, showing how our solution is resilient to churn during the merger, something widely believed to be difficult or impossible. We evaluate the algorithm for various scenarios and show that even when falsely detecting a merger, the algorithm quickly terminates and

does not clutter the network with many messages. The algorithm is flexible since the tradeoff between message complexity and time complexity can be adjusted by a parameter.

7.4 How to implement efficient decentralized transactions over the Internet

We have built a system that implements efficient transactions on a structured overlay network that is subject to churn and Internet-style failures (nodes crashing or communication links behaving intermittently). The transactions are built on top of a key/value store. Previous key/value stores built on structured overlay networks, called Distributed Hash Tables (DHTs), lack support for atomic transactions and strong data consistency among replicas. This is unfortunate, because consistency guarantees and transactions would allow a wide range of additional application domains to benefit from the inherent scalability and fault-tolerance of DHTs.

We have built a key/value store that supports strong data consistency and atomic transactions. It uses an enhanced Paxos commit protocol with only four communication steps rather than six for a commit protocol that uses the basic Paxos algorithm. The improvement was possible by exploiting information from the replica distribution in the DHT. We use symmetric key replication to ensure data availability in the face of churn. We enforce data consistency by using the Paxos algorithm to perform all data operations on a majority of replicas. We have built the Scalaris and Beernet transactional stores that use this algorithm to perform atomic commit. These stores enable implementation of more reliable and scalable infrastructure for collaborative Web services that require strong consistency and atomic changes across multiple items.

7.5 How to do load balancing on structured overlay networks

We have developed a practical load balancing algorithm for structured overlay networks that uses local techniques complemented with estimates of global information obtained with gossiping. One of the biggest impacts on the performance of a key/value store built on a structured overlay network (a Distributed Hash Table or DHT), once it is created, is its ability to balance load among its nodes. DHTs supporting range queries for example suffer from a potentially huge skew in the distribution of their items since techniques such as consistent hashing cannot be applied. Thus explicit load balancing

schemes need to be deployed. Several such schemes have been developed as part of recent research. Most of them use only information locally available in order to scale to arbitrary systems. Gossiping techniques however allow the retrieval of fairly good estimates of global information with low overhead. Such information can then be added to existing load balancing algorithms that can use the additional knowledge to improve their performance.

We have developed several schemes that use global information like the average load and the standard deviation of the load among the nodes to primarily reduce the number of items an algorithm moves to achieve a certain balance. We have extended two novel load balancing algorithms with implementations of those schemes and we have simulated them on several scenarios. Most of these variants show better balance results and move far less items than the algorithms they are based on.

The best of our algorithms achieves a 15-30% better balance and moves only about 50-70% of the number of items its underlying algorithm moves. This variation is also very robust to erroneous estimates and scales linearly with the system size and system load. Further experiments with self-tuning algorithms that set an algorithm's parameter according to the system's state show that even more improvements can be obtained. A variant based on the algorithm described by Karger and Ruhl shows the same balance improvements of 15-30% as the variant above but reduces the number of item movements further to 40-65%.

7.6 How to do simulation and deployment of peer-to-peer systems

We have built a simulation and emulation environment for peer-to-peer systems that also supports deployment. The environment uses the Kompics component framework to achieve a strong modularity property: the same implementation can be deployed on a real system, simulated in reproducible fashion (deterministically), and simulated in real time (nondeterministically, connected to the external world). The environment is released together with the Kompics framework and can be downloaded from `kompics.sics.se`. The environment consists of:

1. a Java-based domain-specific language for specifying peer-to-peer experiment scenarios,

2. a generic simulator for executing Kompics system in a reproducible simulation mode,

3. a generic discrete-event simulator encapsulated in a Kompics component, paired by an orchestrator component for the emulation mode,

4. system-specific simulator components, and

5. component architectures and patterns that enable the execution of Kompics P2P systems in either simulation, in local real-execution/emulation mode, or in distributed deployment.

Several experiment scenarios can be found and followed at the Kompics site. These include scenarios for experiments with Chord and Cyclon in both simulation and real-time execution mode, as well as BitTorrent simulation experiments. The evaluation environment was successfully used as a teaching tool in the Distributed Computing, Peer-to-Peer and Grids course (ID2210) at KTH. The framework was used as support for student assignments which required the implementation and evaluation of P2P systems including a structured overlay network, a gossip-based overlay, and a content distribution network.

7.7 How to make peer-to-peer networks secure

Securing a structured overlay network is difficult. Structured overlay networks are based on a specific topology and rely on the assumption that most of the nodes are non-malicious so that this topology is not disturbed. The network is vulnerable to a Sybil-type attack because it gives the attacker the opportunity to control many nodes. Furthermore, the SON requires maintenance to preserve its particular topology. This maintenance is vulnerable to various attacks, such as the Eclipse attack, and attacks that exploit network failures and churn. Given these facts, what can we do to achieve security? One approach is to use another topology, such as a small-world network (SWN). Small-world networks are closely related to social networks, which have trust and identity relationships which strongly reduce the possibility of Sybil-type attacks. Link maintenance is also cheaper than with a SON because the links are much more static in a SWN. The links are also more robust because they have a stochastic element that makes the network somewhere inbetweeen a structured network and a random one. The random properties make it harder to disconnect a SWN during churn.

Given the good security properties of a SWN, can we use a SWN instead of a SON for our self-managing transactional store? This is a nontrivial question, since many SWN topologies do not allow efficient routing. We have found that the answer to this question is *yes*. We can design the SWN to be self-organizing and to achieve efficient routing. A small-world network is

practical for routing in a fault-tolerant setting: it can do successful routing (>90% success) in the presence of 60% of failed nodes. It is tolerant to network partitioning: within each partition, there is almost no ill effect from the partition (less than 0.02% routing failures). We conclude that future decentralized service architectures can use small-world network topologies to achieve security.

8 Partners and Contact Information

We list the seven partner institutions and we give contact information for the partner leaders. The SELFMAN project's official website is `www.ist-selfman.org` and the coordinator is Peter Van Roy at UCL.

8.1 Université catholique de Louvain (UCL)

Prof. Peter Van Roy (project coordinator)
(home page `www.info.ucl.ac.be/~pvr`)
Dept. of Computing Science and Engineering
Place Sainte-Barbe, 2
B-1348 Louvain-la-Neuve, Belgium
Phone: (++32) 10 47 83 74 (Secretary: (++32) 10 47 31 50)
Cellular: (++32) 485 42 46 77
Fax: (++32) 10 45 03 45
E-mail: `peter.vanroy@uclouvain.be`

8.2 Royal Institute of Technology (Kungliga Tekniska Högskolan) (KTH)

Prof. Seif Haridi (home page `www.sics.se/~seif`)
Department of Microelectronics and Information Technology
Box 1263
S-164 28 Kista, Sweden
Phone: (++46) 8 6331530
Cellular: (++46) 70 512 1540
Fax: (++46) 8 7517230
E-mail: `seif@it.kth.se`

8.3 Institut National de Recherche en Informatique et Automatique (INRIA)

Prof. Jean-Bernard Stefani
(home page www.inria.fr/personnel/Jean-Bernard.Stefani.fr.html)
INRIA Rhône-Alpes, Inovallée
655 Avenue de l'Europe, Montbonnot
F-38334 Saint Ismier cedex, France
Phone: (++33) 4 76 61 52 57 (Secretary: (++33) 4 76 61 52 59)
Fax: (++33) 4 76 61 52 52
E-mail: Jean-Bernard.Stefani@inria.fr

8.4 France Telecom Research and Development

Dr. Thierry Coupaye
France Telecom R&D/MAPS
28 Chemin du Vieux Chêne, BP98
F-38243 Meylan, France
Phone : (++33) 4 76 76 43 21
Fax: (++33) 4 76 76 45 57
E-mail: Thierry.Coupaye@orange-ft.com

8.5 Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB)

Prof. Alexander Reinefeld (home page www.zib.de/reinefeld)
Zuse Institute Berlin
Takustr. 7
D-14195 Berlin-Dahlem, Germany
Phone: (++49) 30 84185 130
Fax : (++49) 30 84185 311
Email: ar@zib.de

8.6 Peerialism AB

Johan Ljungberg
Apelvägen 28
S-18275 Stocksund, Sweden
Email: johan@peertv.se
Cellular: (++46) 70 870 4928

8.7 National University of Singapore (NUS)

Prof. Roland Yap (home page www.comp.nus.edu.sg/~ryap)
Department of Computer Science
Office: S16-06-18
School of Computing
National University of Singapore
S16 Level 5, 3 Science Drive 2
Singapore 117543, Republic of Singapore
Phone: (++65) 6516-2972
Fax: (++65) 6779-4580
E-mail: ryap@comp.nus.edu.sg