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

Periodic Activity Report - Year Two

Due date of deliverable:	July 15, 2008
Actual submission date:	July 15, 2008
Start date of project:	June 1, 2006
Duration:	36 months
Dissemination level:	CO

1 Executive summary

With distributed systems growing larger and more complex, it is increasingly clear that managing them is beyond conventional technologies. The SELFMAN project proposes a solution: a scalable self-managing application infrastructure. In the project's second year we have implemented the first scalable transaction system for the Internet. Our system is built on top of a structured peer-to-peer network. It provides a reliable transactional store despite Internet failures (both permanent node failures and temporary network failures). To achieve this result, we have designed, simulated, and implemented several innovative algorithms using innovative programming techniques. We have developed an algorithm that implements atomic commit in the hostile Internet environment and an algorithm that can gracefully survive network partitions (by splitting and merging). The atomic commit algorithm uses the Paxos uniform consensus protocol.

We are focusing our resources on two applications: a Distributed Wiki and a media streaming application. We have used the first implementation of our scalable transaction infrastructure to build a prototype application, a Distributed Wiki. This application won first prize in the first IEEE International Scalable Computing Challenge (SCALE 2008).

The media streaming application is being developed by a new project partner, Peerialism AB, who joined the project in the beginning of the second year. Peerialism AB (formerly Stakk AB) is a Swedish start-up company in the area of dynamic media streaming. Peerialism is implementing a media streaming application as its first product. In the second year, significant parts of this application have been built and tested. Hard real-world problems, such as bypassing Network Address Translation (NATs), have been overcome. Peerialism is using a SELFMAN structured overlay network and the SELFMAN component model, Kompics, for its dynamic reconfiguration challenges.

In addition to these main results, SELFMAN also delivered a series of other significant results, which are important to support the main results and to explore the area around them. These results include the release of the Mozart programming system version 1.4.0, which provides network-transparent distribution with support for asynchronous failure detection and wide choice of distributed protocols to implement language entities, all in a powerful concurrent symbolic language. We have also made complete implementations of two structured overlay networks, DKS and P2PS, an advanced component model, Kompics, and advanced tools for distributed deployment and monitoring of dynamic software architectures, FructOz and LactOz.

2 Project objectives and major achievements

The overall objective of SELFMAN is to address directly the challenge of building large distributed systems. The approach taken by SELFMAN is to build and prove the effectiveness of an application infrastructure for scalable self-managing applications based on extending a structured overlay network with self-* services and a transaction protocol.

We briefly recapitulate the project's first year. In the first year, we built two structured overlay networks (DKS and P2PS) and we explored the programming model and component architecture for expressing the self-management operations. We also defined a transaction protocol and collected four application scenarios.

In the second year, we completed most of the groundwork. We implemented the transaction algorithm and a network merge algorithm for handling partitions. We released the Mozart-1.4.0 system, with support for asynchronous failure handling and with choice of protocols for distribution of language entities. We designed the Kompics model and we are porting DKS to Kompics. We built two tools, FructOz and LactOz, for deployment and monitoring of dynamic distributed software architectures. We focused our application development on two applications, the Distributed Wiki and the media streaming application, and we implemented significant parts of both applications. Section 2.1 explains all these achievements in more detail.

2.1 Major achievements in the second year

In the second year, we have achieved the following main results:

- **Structured overlay networks**. We completed our two implementations, DKS and P2PS, of structured overlay networks. These implementations explore different approaches: static versus dynamic, highly concurrent versus less concurrent system, symbolic versus non-symbolic language, synchronous versus asynchronous failure detection. The interchange between these two implementation philosophies is porting fruit: for example, the Kompics component model implemented in Java takes many ideas from the Oz approach. The "channel" concept in Kompics is inspired by the "port" concept in Oz.

- **Components**. We completed the two major parts of our component research for distributed computing. First, we designed and implemented a component model, Kompics, that gives a defensible semantics for concurrency and partial failure in reconfigurable distributed

systems. We have tested Kompics under small scenarios, DKS is being reimplemented in Kompics, and the Peerialism product is using Kompics. Second, we designed a component model for distributed self-configuration, based on two tools, FructOz and LactOz, that allow to deploy and monitor dynamic software architectures. FructOz is a library for deploying and configuring and LactOz is a library for navigating and monitoring. These two tools are the actuating and monitoring agents in the self-configuration feedback loop. Both libraries extensively use the symbolic computation and lightweight concurrency abilities of Oz. Finally, we are studying dynamic transactional reconfiguration in an implementation of the Fractal component model.

- **Transactions**. We implemented the transaction protocol over structured overlay networks and implemented it in a prototype Distributed Wiki application (which has won a prize, see Section 5.1). The protocol is based on the Paxos uniform consensus algorithm for achieving atomic commit. Both in the Kompics model and the Mozart 1.4.0 distribution model, the implementation of this algorithm is very close to its high-level specification as given by Guerraoui and Rodrigues [2]. Note that even without churn, imperfect failure detection can create inconsistencies in distributed hash tables based on structured overlay networks. We studied these inconsistencies and presented solutions for reducing their probability. The network merge algorithm plays an important role in managing the inconsistency.

- **Network partitioning**. We designed an algorithm for self-healing in the case of network partitioning. We have made first implementations of this algorithm and we have extensively simulated and studied it. The heart of the algorithm is a network merge: it detects when the network partition is over and merges two overlay networks into a single network.

- **Applications**. In the second year, we decided to focus our resources on two application scenarios: Distributed Wiki (by ZIB) and media streaming application (by Peerialism). In the first year, we originally had four application scenarios. The J2EE application server (from Bull) was quickly dropped because it was not Internet-based. In the second year, we have dropped the M2M messenging application of France Télécom, because of limited resources. We are now focusing on the two remaining scenarios. The media streaming application will become a product of Peerialism. The Distributed Wiki has already won a prize

(see before). The Distributed Wiki application is written in Erlang, a programming system with very similar approach as Mozart.

- **Mozart 1.4.0 Programming System**. We completed and made a public release of the Mozart 1.4.0 system. This system supports flexible network-transparent distribution (including distributed garbage collection) with asynchronous failure handling. The design, implementation, and semantics of this system are explained in the Ph.D. dissertation of Raphaël Collet [1] (see Appendix of Deliverables document). It is the most advanced language we know of for robust distributed programming. This system cross-cuts the workpackages: P2PS is implemented in Mozart 1.4.0 (WP1), FructOz and LactOz are being ported to Mozart 1.4.0 (WP2), the transaction protocol and the merge protocol (WP3 and WP4) are being ported to Mozart 1.4.0.

2.2 Other results

In addition to these major results, we have a series of other results that fit into our overall goal of building self-managing applications:

- **Security over SONs**. We studied a simple model for security over structured overlay networks. SONs are inherently difficult to secure because of their structure. We are studying variations of this structure, small-world networks, that are more appropriate for security mechanisms.

- **Live streaming**. We developed live streaming mechanisms over a structured overlay network and tested them extensively in the SicsSim-B simulator.

3 Workpackage progress of the period

This section gives for each workpackage the objectives, the progress made during the period, the (eventual) deviations and corrective actions, the deliverables, and the milestones. The progress is broken down per partner and a brief explanation of each partner's work is given.

The information of this section is summarized in Tables 1 and 2. Table 1 gives the project milestones for the second year, their due dates and their actual delivery date. Table 2 gives the project deliverables for the second year, their actual delivery dates and their estimated and used person-months. (Note that Table 3 in Section 4 gives a detailed breakdown of each workpackage per partner.)

Milest. no.	**Milestone name**	**WP no.**	**Date due**	**Actual delivery date**	**Lead contractor**
M1.2	Finished structured overlay network with component model	WP1	M24	M24	KTH
M2.2	Finished architectural framework with component model using SON	WP2	M24	M24	UCL
M3.2	Finished replicated storage service over structured overlay network	WP3	M24	M24	ZIB
M3.3	Finished query layer for replicated storage service	WP3	M24	M24	ZIB
M4.1	Understand how to incorporate self-* services on architectural framework	WP4	M24	M24	INRIA
M5.2	Understand application structure for application scenarios	WP5	M24	M24	FT
M6.2	Possible synergies and collaborations are realized	WP6	M24	M24	UCL

Tab. 1: Milestones list for second year (with delays)

Del. no.	Deliverable name	WP no.	Date due	Actual date	Est. indic.	Used indic.	Lead contr.
D1.2	Rep. on high-level self-man. primitives for SON	WP1	M24	M24	16	12.5	UCL
D1.3b	Final rep. on secur. for SON	WP1	M24	M24	5	11.5	NUS
D1.4	Java library of SELFMAN SON (Software)	WP1	M24	M24	4	8	KTH
D1.5	Mozart library of SELF-MAN SON (Software)	WP1	M24	M24	8	2	UCL
D2.1b	Rep. on computation model with self-man. primitives	WP2	M24	M24	6	10	KTH
D2.1c	Component-based computation model (Software)	WP2	M24	M24	12	21.56	KTH
D2.2b	Rep. on architectural framework tool support	WP2	M24	M24	9	9	PR
D2.2c	Architectural framework (Software)	WP2	M24	M24	19	5	INRIA
D2.3b	Rep. on formal oper. semantics (distr. abstractions)	WP2	M24	M24	5	3	INRIA
D3.1b	2nd rep. on formal models for transactions over SON	WP3	M24	M24	3	3.33	KTH/ZIB
D3.2a	Replicated storage service over a SON (Software)	WP3	M24	M24	18	3	ZIB
D3.3a	Simple database query layer for replicated storage service (Software)	WP3	M24	M24	8	3	ZIB
D4.1a	1st rep. on self-config. supp.	WP4	M24	M24	4	6.5	INRIA
D4.2a	1st rep. on self-healing supp.	WP4	M24	M24	3	9.7	UCL
D4.3a	1st rep. on self-tuning supp.	WP4	M24	M24	5	2	ZIB
D4.4a	1st rep. on self-prot. supp.	WP4	M24	M24	10	11	NUS
D5.2a	Design specification of self-managing application	WP5	M24	M24	6	7	FT
D6.1c	2nd-yr. project workshop	WP6	M24	M24	1	1	UCL
D6.5b	2nd progress and assessment rep. with lessons learned	WP6	M24	M24	3	2	UCL

Tab. 2: Deliverables list for second year (with delays)

3.1 Workpackage 1: Structured overlay network and basic mechanisms

Objectives To build two structured overlay networks, one based on mainstream technology (Java) and another based on an asynchronous dataflow language (Mozart), and release them as finished software packages.

Progress KTH has completed development of DKS, which is now completely operational. KTH has also developed live streaming mechanisms and extensively tested them in the SicsSim-B simulator, as a part of the work with Peerialism and their media streaming application.

UCL has completed development of P2PS, which is now operational under the Mozart 1.3.2 distribution model. We are now porting it to the new Mozart 1.4.0 model, which should result in significant simplification and improved architecture. P2PS is based on the relaxed ring structure, which greatly reduces probability of lookup inconsistency. We have verified this through simulation. We have implemented the CiNiSMO tool, a network simulator for P2PS and the PEPINO visualization tool for P2PS.

Peerialism worked on issues related to building a structured overlay network when there is NAT (Network Address Translation) in the network. The NAT causes different parts of the network to have different IP addresses, which are mapped when the NAT gateway is crossed. The problem is knowing, for all combinations of NAT types, which ones can connect to each other. This is a low-level issue, but it is essential for Peerialism's media streaming application.

NUS continued their work on security for structured overlay networks. Because of the late start in the first year, we continued in the second year to work on some issues which were postponed in the first year. We identified in D1.3a the importance of identity and network maintenance. We find that SONs are inherently difficult to secure because of their structure. We find that a slightly different network structure, based on social networks and small-world networks (SWN), is a way of mitigating these issues. We built a prototype testbed for investigating the use of SWN as a self-organizing network. Initial results show that SWN routing works well and can be a viable replacement for a SON. The work on SWN is required for the work on self protection in WP4. NUS also worked on a prototype software component authentication system which can check software components against their signatures. This system helps in determining the origin and version of all software components. This is important for self configuration when components come from different sources. NUS also worked on completing the survey that was started in the first year and on a monitoring infrastructure

which can determine if a collection of processes is behaving properly. We have used our monitoring infrastructure to study the behavior of Skype.

Deviations and corrective actions Peerialism's work on NAT belongs in the first year (it is a low-level issue) but they only joined the project in the second year. Solving the NAT issue is an essential problem for their media streaming application. NUS required more effort than predicted for the SWN infrastructure, since it had to be built from scratch. We reduced some person-months in WP1 but supplemented it with support from NUS. We also used more effort than anticipated for implementing our component authentication system, which is implemented in Windows, because of the complexity of the OS and its closed nature.

Deliverables D1.2 (Report on high-level self-management primitives for SON), D1.3b (Final report on security for SON), D1.4 (Java library of SELFMAN structured overlay network - Software), D1.5 (Mozart library of SELFMAN structured overlay network - Software)

Milestones M1.2 (Finished structured overlay network with component model)

3.2 Workpackage 2: Service architecture and component model

Objectives To build a distributed component architecture that has the basic primitives needed for self management.

Progress UCL has completed the Mozart 1.4.0 system (see `www.mozart-oz.org`) and released it as the next major update of Mozart under an open source license. Mozart 1.4.0 implements completely the distribution and fault models presented in Raphaël Collet's Ph.D. dissertation (see Appendix of Deliverables document). This system is the foundation for UCL's structured overlay network and component model implementations. The system has an asynchronous failure-handling model based on fault streams, that generalizes Erlang's failure model for imperfect failure detection, compositional distribution abstractions, and multiple distributed protocols for network-transparent distribution.

KTH has designed and implemented a reactive component model for distributed computing, known as Kompics, in tight discussion with INRIA and

France Télécom. France Télécom and INRIA have developed several practical applications and middleware on clusters using the Fractal model; Kompics complements the Fractal model with an event-based execution model, which can be used for the construction of communication-intensive software. KTH, France Télécom, and INRIA have worked together to tightly integrate the Kompics Java implementation and the Fractal Java reference implementation, Julia. The design for this integration has been worked out–as reported in deliverable D2.1b. Future work will realize the integration, so that the Kompics execution model can be used by Fractal developers and Kompics developers can benefit and make use of the Fractal toolset. Kompics is currently tested under small scenarios and several generic components needed for distributed applications have been implemented in it. DKS is being ported to the Kompics model.

INRIA has developed tools for distributed deployment in Mozart, that provide the mechanisms for self-configuration support. This work overlaps with WP4: the tools use a component model in order to support self-configuration. The tools, FructOz and LactOz, are further described in WP4. INRIA has also developed a formal specification of the Fractal component model using the Alloy specification language, reported in deliverable D2.1b. In cooperation with KTH and France Télécom, INRIA has started work on a formal specification of the Kompics model in Ally, which refines the Fractal one.

Peerialism worked on an application-level network and concurrency emulator "MyP2PWorld", which is an important tool for testing their structured overlay network. It has the three main properties of reproducibility, simplicity of deployment, and ability to target production code. Peerialism also worked with the Kompics team to re-architect part of their network software, a tracker, to use the event-driven component model of Kompics. Initial results show performance improvement in addition to cleaner and more maintainable architecture.

Deviations and corrective actions UCL's work on the Mozart 1.4.0 system to a sufficient level of robustness took more effort than initially estimated, but we made a successful release just after the end of the second year. Peerialism moved several person-months from WP3 in order to work on its testing tool in WP2.

Deliverables D2.1b (Report on computation model with self-management primitives), D2.1c (Component-based computation model - Software), D2.2b (Report on architectural framework tool support), D2.2c (Architectural frame-

work - Software), D2.3b (Report on formal operational semantics - distributed abstractions)

Milestones M2.2 (Finished architectural framework with component model using SON)

3.3 Workpackage 3: Self-managing storage and transactions

Objectives To complete the transaction algorithm and its replicated storage over a structured overlay network and to make a first implementation.

Progress KTH and ZIB have completed the transaction algorithm. Even without churn, imperfect failure detection can create inconsistencies, so the algorithm must be carefully designed to support this. KTH has attacked this problem both on the routing level and on the data level. On the routing level, simple changes to the atomic join and leave protocols can significantly reduce the probability of inconsistency. On the data level, we use the Paxos uniform consensus algorithm for atomic commit. This algorithm handles the imperfect failure detection of the Internet by using a majority for consensus.

UCL has made a first implementation of the transaction algorithm over P2PS.

Deviations and corrective actions There were no significant deviations in the project's second year.

Deliverables D3.1b (Second report on formal models for transactions over structured overlay networks), D3.2a (Replicated storage service over a structured overlay network - Software), D3.3a (Simple database query layer for replicated storage service - Software)

Milestones M3.2 (Finished replicated storage service over structured overlay network), M3.3 (Finished query layer for replicated storage service)

3.4 Workpackage 4: Self-management services

Objectives To design and implement the self-management services needed by our applications.

Progress KTH has developed and simulated an algorithm for handling network partitioning. The key of the algorithm is the network merge: when nodes detect a network partition, a merge algorithm is initiated. This work is accepted for journal publication. It is the first ever practical solution to network partitioning for structured overlay networks (and goes counter to much of the "folk wisdom" in the area).

INRIA has developed a framework for support self-configuration in Mozart. There are two tools. The first, FructOz, is a library that supports deploying and configuring complex dynamic software architectures. The second, LactOz, is a library that supports navigating and monitoring in dynamic software architectures. Together, the two tools provide the actuating and monitoring agents in the self-configuration feedback loop, and allow the construction of distributed components that integrate such a feedback loop in their implementation. Future work will integrate transactional aspects in the tools, drawing from the work carried out in France Télécom on transactional reconfiguration and on ZIB and KTH work on transactions on SONs. INRIA has also worked on self-repair aspects. Initial findings and requirements on a component model supporting self-repair have been identified. A prototype for self-repair in cluster-size systems is being completed, and will be finalized, evaluated and reported on in the third year.

France Télécom has developed a new implementation of the Fractal model in Java that provides transactional dynamic reconfiguration. This complements the work by INRIA. France Télécom has also worked on self tuning services: specification of a model for performance characterization of black-box components based on load injection and further evolution of the CLIF load injection framework. The CLIF framework will be used in the third year as part of the evaluation of the SELFMAN applications.

NUS is looking at self-protection mechanisms for SWNs, which are apparently much easier to secure than SONs. The security weaknesses of SONs seem to be intrinsic to how SONs such as DHTs are designed. We therefore are looking at somewhat different kinds of self-organizing networks, social and SWNs, which can be easier to secure. Our first conclusions show that a SWN may be a feasible replacement for a SON when there are malicious nodes or users. We have investigated some forms of attack and some initial security mechanisms to defend against these attacks.

Deviations and corrective actions UCL did not have the time to implement the merge algorithm in P2PS. This will be done in the beginning of the third year of the project. NUS used more person-months than anticipated because they are investigating both the proposal of replacing a SON by a

SWN as well as the security issues of SWNs. We have supplemented the project support with other support from NUS.

Deliverables D4.1a (First report on self-configuration support), D4.2a (First report on self-healing support), D4.3a (First report on self-tuning support), D4.4a (First report on self-protection support),

Milestones M4.1 (Understand how to incorporate self-* services on architectural framework

3.5 Workpackage 5: Application requirements and evaluations

Objectives To specify and build self-managing applications according to application scenarios that are interesting for the industrial partners.

Progress ZIB, with help from KTH, was able to implement the transaction algorithm in a prototype Distributed Wiki. This application was implemented in Erlang, a system with support for transparent distribution based on asynchronous failure detection (similar to a subset of the new Mozart 1.4.0 system).

Peerialism is developing its media streaming application, in tight collaboration with KTH, especially regarding the streaming protocols on top of structured overlay networks and the component model. Peerialism is using the Kompics model in its application, to handle the dynamic updates and changes of structure that are needed. Peerialism re-architected part of their network software to use this model (this work overlaps with Workpackage 2, q.v.).

France Télécom has defined a more formal definition of the M2M application (as requested during the first year review) and has completed the D5.1 and D5.2a deliverables on use requirements and design specifications of the SELFMAN applications. Two applications have been kept in D5.2a: the Distributed Wiki and the media streaming application.

Deviations and corrective actions We decided not to continue development on the M2M application specified by France Télécom. This application exists in another form inside of France Télécom, but we do not have resources to develop it within SELFMAN (in particular, France Télécom is not sufficiently funded in SELFMAN to do this application). France Télécom will return to its initial role (defined in Description of Work) as a provider

of realistic simulation data and evaluator. We will limit our work in SELFMAN to the two applications, the Distributed Wiki and the media streaming application.

Deliverables D5.2a (Design specification of self-managing application)

Milestones M5.2 (Understand application structure for application scenarios)

3.6 Workpackage 6: Management, dissemination, and exploitation

Objectives To manage the project, disseminate results, and collaborate with other projects.

Progress We have successfully managed the departure of ePlus and the incorporation of Peerialism (formerly called Stakk) as new industrial partner starting from the beginning of the second year. We have organized five project meetings, in addition to numerous visits of individual researchers among partners and to non-SELFMAN institutes for collaboration. This has resulted in several papers between SELFMAN partners and non-SELFMAN researchers, on SELFMAN-related topics. KTH has collaborated with SICS to make a Kompics poster for joint internal and external dissemination. France Télécom has guest-edited a special issue of the journal Annals of Telecommunications on software components.

In collaboration with the Grid4All project, we are organizing the *Workshop on Decentralized Self Management for Grids, P2P, and User Communities* as an official SASO 2008 workshop (see Section 5.2).

We wrote a three-page dissemination article, *A self-managing peer-to-peer network*, in the magazine eStrategies Projects published by British Publishers in June 2008 (see Section 5.3). This article will be sent in 39,000 copies to decision makers in the public and private sectors throughout Europe and neighboring regions.

Our Distributed Wiki application (written by ZIB in collaboration with KTH), *A Transactional Scalable Distributed Data Store: Wikipedia on a DHT*, won first prize in the First IEEE International Scalable Computing Challenge (SCALE 2008), Lyon, France, May 2008 [3]. This prize comes with a plaque and a monetary reward (see Section 5.1).

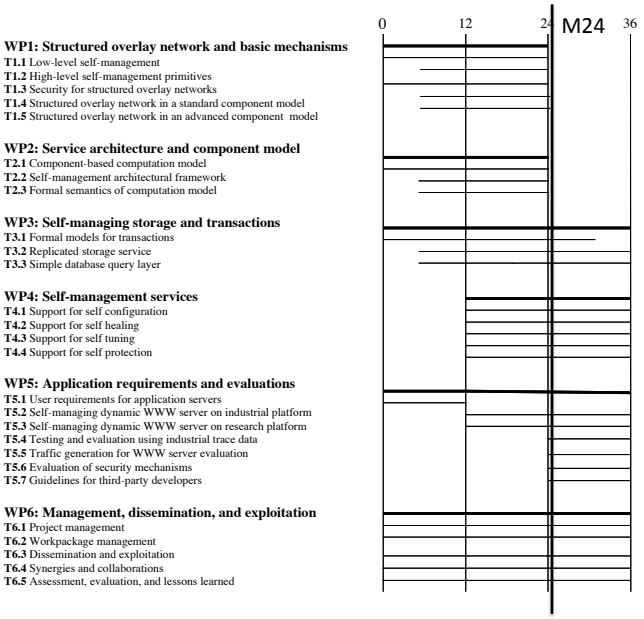


Fig. 1: Project timetable with status line after second year

Deviations and corrective actions There were no significant deviations in the project's second year.

Deliverables D6.1c (Second project workshop), D6.5b (Second progress and assessment report with lessons learned)

Milestones M6.2 (Possible synergies and collaborations are realized)

4 Consortium management

We give a summary of the project status, its management and follow-up activities. Figure 1 shows the project timetable after the second year. In the second year, the partner ePlus left the project and a new partner, Peerialism (formerly called Stakk), joined the project. We have made a revised Description of Work dated February 27, 2008 that takes this change into account. We maintain our project goals.

Table 3 gives a detailed breakdown of the person-months spent by each partner in the second year, including the estimated own contributions of the AC partners. Note that ePlus remains in the table (sixth partner, called EP) with '0' in all columns and Peerialism (eighth partner, called PR) is added.

Person-Month Status Table																
			TOT	UCL	KTH	INR.	FT	ZIB	EP	NUS	PR	**ACT**	UCL	KTH	ZIB	NUS
WP1	Structured overlay network and basic mechanisms	**Act tot**:	**34**	2	10	3.5	0	6	0	11.5	1	**0**	4	0	1	1.5
		Plan tot:	**51**	14	10	8	0	0	0	17	2	**0**				
WP2	Service architecture and component model	**Act tot**:	**48.56**	12	21.56	7	1	4	0	0	3	**0**	4	2	0	0
		Plan tot:	**70**	17	12	19	11	11	0	0	0	**0**				
WP3	Self-managing storage and transactions	**Act tot**:	**9.33**	1	6.33	0	0	2	0	0	0	**0**	2	0.5	1	0
		Plan tot:	**43**	6	11	0	8	15	0	0	3	**0**				
WP4	Self-management services	**Act tot**:	**29.2**	0	4.7	6.5	5	2	0	11	0	**0**	0	0	0	1.5
		Plan tot:	**76**	6	14	19	3	15	0	17	2	**0**				
WP5	Application requirements and evaluations	**Act tot**:	**7**	0	0	0	4	2	0	0	1	**0**	0	0	1	0
		Plan tot:	**47**	9	5	6	9	11	0	3	4	**0**				
WP6	Management, dissemination, and exploitation	**Act tot**:	**3.9**	1	0	0.4	1	0	0	0.5	1	**0**	4	0.5	0	0.5
		Plan tot:	**30**	10	3	3	5	3	0	3	3	**0**				
Total project person-month		**Act tot**:	**132.49**	16	42.59	17.4	11	16	0	23.5	6	**0**	14	3	3	3.5
		Plan tot:	**317**	62	55	55	36	55	0	40	14	**0**	0	0	0	0

Estimation of AC own personal cost		
AC institution	Cost/month	Total cost for period
UCL	3*6250/month	18750
	10*4600/month	46000
	1*4000/month	4000
KTH	3*10500/month	31500
ZIB	3*8500/month	25500
NUS	2*1522/month	3044
	1.5*6500/month	9300

Tab. 3: Person-month status table (contract no. 34084, acronym SELFMAN, period 1/6/2007 to 31/5/2008)

4.1 Consortium update

4.1.1 Stakk/Peerialism

The original partner ePlus has formally left the project and Stakk (now renamed Peerialism) has officially joined the project starting from the beginning of the second year. The paperwork took longer than planned, because of delays both at UCL and at the Commission, but the official acceptance letter arrived before the end of May 2008.

4.1.2 Application scenarios: from one to four to three to two

We clarify the role of the application scenarios in the project. The initial Description of Work proposed one application scenario. In the project's first year, we extended this to propose three application scenarios: a M2M application (France Télécom), a Distributed Wiki (ZIB), and a media streaming product (Stakk/Peerialism). (Initially, there was a fourth application scenario, a J2EE application server proposed by Bull, but we dropped this since it runs on a cluster and not on Internet.) This gave us a wider coverage of the space of peer-to-peer applications. In the second year of the project, we selected two of the three remaining applications, the Distributed Wiki and the media streaming product, to be implemented. We dropped the M2M application proposed by France Télécom. France Télécom does not have the resources in the project to develop the M2M application, so we decided to focus on the two most promising applications for which we do have resources.

To develop the two chosen applications, in addition to SELFMAN resources, we are relying on own resources of ZIB and Peerialism. ZIB has developed a prototype Distributed Wiki that has won a prize (see Section 5.1) and Peerialism is developing its application into a product. France Télécom will play its original project role of initiator and evaluator. In the third year, it will be an evaluator. It will coordinate the evaluation of the two applications including their self-* abilities.

4.2 Coordination activities

In the second year, we have the following coordination activities at the level of the SELFMAN project (in addition to local collaborations done by individual partners):

- We continued our collaboration with the Grid4All project. We submitted a joint workshop proposal to EuroSys 2008, which was rejected because there were too many submissions. We then submitted a joint workshop proposal to SASO 2008, which was accepted (see Section 5.2).

- We continued our collaboration through the CoreGRID Network of Excellence, of which four SELFMAN partners are also partners. We actively submit and present our work at CoreGRID events and publications.

4.3 Project meetings

We organized the following project meetings during the second year of the project:

- Project meeting in Stockholm (KTH), June 5-7, 2007. Meeting between KTH and ZIB on self-managing routing tables in DHTs.
- Project meeting in Stockholm (KTH), June 18-22, 2007. Meeting between KTH and ZIB on transactions in structured overlay networks.
- Project meeting in Grenoble (INRIA and FT), Nov. 20-23, 2007. This meeting discussed project progress and plans for collaboration in component models, the transaction algorithm, the applications. We also discussed in detail the responsibilities for the work to be done in the second year and the deliverables that this work is part of. Participants were from INRIA, France Télécom, KTH, ZIB, Peerialism, and NUS. The SELFMAN Wiki has meeting notes.
- Project meeting in Paris (France Télécom building in Paris), Mar. 27-28, 2008. This meeting was focused on discussions regarding the Fractal and Kompics component models.
- Project meeting in Berlin (ZIB), May 22-23, 2008. This meeting presented five technical talks giving second-year results of the project and discussed the responsabilities for the various deliverables. The talks were on component models (Kompics and distributed deployment in Oz), small-world networks and self protection, the Wikipedia on DHT application, and a study of data consistency in structured overlay networks. Roland Yap and Sameh El-Ansary participated through Skype. Roland Yap gave a talk with interactive questions and answers, which went well because of the reliable Internet link between Germany and Singapore. Participants were from INRIA, UCL, KTH, ZIB, NUS, and Peerialism. The SELFMAN Wiki has meeting notes and talk slides.

In addition to these meetings, numerous visits were done by individual researchers between SELFMAN partners. Many of these individual visits were funded by the CoreGRID network of excellence (UCL, KTH, ZIB, and INRIA are CoreGRID partners).

5 Plan for using and disseminating the knowledge

We are organizing a workshop on self management that will be held together with a major international conference, SASO 2008 (see Section 5.2). We wrote a three-page article that will be distributed to decision makers all over Europe (see Section 5.3).

5.1 Distributed Wiki wins a prize

An unexpected result of the second year was that our Distributed Wiki application, *A Transactional Scalable Distributed Data Store: Wikipedia on a DHT*, won first prize in the First IEEE International Scalable Computing Challenge (SCALE 2008), Lyon, France, May 2008 [3]. This prize comes with a plaque and a monetary reward. The application was written by ZIB with help from KTH and uses the application infrastructure developed in SELFMAN: a transaction algorithm with replicated storage running over a structured overlay network.

5.2 Project workshop

We are organizing the *Workshop on Decentralized Self Management for Grids, P2P, and User Communities*:

`http://www.ist-selfman.org/wiki/index.php/SelfmanWorkshop`

which will be held in conjunction with the Second International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2008):

`http://polaris.ing.unimo.it/saso2008`

on Oct. 20-24, 2008. We advertised the workshop widely on mailing lists and internal lists. The Workshop Call for Papers is reproduced in Deliverable D6.1c (see Deliverables document).

The workshop is organized by SELFMAN together with the Grid4All project, with corporate sponsorship by France Télécom Research and Development. The workshop organizing committee consists of Peter Van Roy (SELFMAN), Marc Shapiro (Grid4All), and Seif Haridi (SELFMAN and Grid4All).

5.3 Article in eStrategies Projects Magazine

SELFMAN wrote and contracted for the publication of a three-page dissemination article, *A self-managing peer-to-peer network*, in the magazine

eStrategies Projects published by British Publishers in June 2008. The article is reproduced in Section 7. This magazine is disseminated to 39,000 decision makers in more than 33 countries in Europe and neighboring regions (including Eastern Europe, CIS, Turkey, Switzerland), to both public and private sectors. This article is free from copyright restrictions and we are also using it internally for dissemination.

The article summarizes the main contribution of the SELFMAN project at the end of the second year. We explain how we extend structured overlay networks with a transaction algorithm and a network partitioning (merge) algorithm. We explain the importance of feedback loops and component models and how we use them. We explain the two applications, the Distributed Wiki and the Peerialism media streaming product, and how they use the technology developed in SELFMAN.

6 Papers and publications

This section lists the papers that were funded by SELFMAN. Most of the papers were completely funded by SELFMAN, unless otherwise indicated.

6.1 Invited talks

- Peter Van Roy. *Reflections on Self Management in Software Development.* Invited lecture at Almende Summer School in agent architecture, multi-agent systems, and self organization, Rotterdam, the Netherlands, Aug. 29, 2007. At: summerschool.almende.com. Also given at Scottish Programming Language Seminar, James Clerk Maxwell's house, Edinburgh, UK, Sept. 28, 2007.

- Peter Van Roy. *Self Management for Large-Scale Distributed Systems.* Invited talk at Grid@Mons 2008. At: grid.umh.ac.be. May 2008.

- Seif Haridi. *Kompics: Reactive Component Model for Distributed Computing.* Invited talk at Grid@Mons 2008, May 2008. At: grid.umh.ac.be.

- Bruno Dillenseger. *The CLIF load testing platform and its utilization at Orange Labs.* In TAROT (Training And Research On Testing) Summer School, Grenoble, July 3 2007.

- B. Dillenseger and M. Marche. *Test management and load testing with Salom-TMF and CLIF.* OW2 Tech Day, Grenoble, May 15, 2008.

- Boris Mejias. *Self-Management of Large Scale Distributed Systems.* Invited talk at DCC investiga - Department of Computer Science of the Pontificia Universidad Catlica de Chile. Santiago, Chile, November 19, 2007. Also given at Department of Computer Science of the Universidad de Chile. Santiago, Chile.

- Thorsten Schütt. *Scalable Wikipedia with Erlang.* Google Scalability Conference, Seattle, June 14, 2008.

- Alexander Reinefeld. *Building a transactional distributed data store with Erlang.* Erlang Exchange 2008, June 26-27, 2008, London.

- Peter Van Roy. *The Challenges and Opportunities of Multiple Processors: Why Multi-Core Processors are Easy and Internet is Hard.* Panel on Reinventing Audio and Music Computation for Many-Core Processors, at the International Computer Music Conference (ICMC 2008), Belfast, Ireland, Aug. 24-29, 2008. Two-page position statement in the conference proceedings.

6.2 Journal papers

- Thorsten Schütt, Florian Schintke, and Alexander Reinefeld. *Range Queries on Structured Overlay Networks.* Journal on Computer Communications, vol. 31, no 2, Elsevier, pp 280-291, February 2008.

- Tallat M. Shafaat, Ali Ghodsi, and Seif Haridi. *Dealing with Network Partitions in Structured Overlay Networks.* Submitted to the Journal of Peer-to-Peer Networking and Applications, Springer-Verlag, 2008. Accepted with revisions. First revision submitted, waiting for further comments by reviewers.

- A. Diaconescu and B. Dillenseger. *Composite Probes: a Generic Monitoring Framework for Hierarchical Data-Processing.* Submitted for journal publication in November 2007. This work is partially funded by SELFMAN.

- P.-C. David, T. Ledoux, M. Léger, and T. Coupaye. *FPath & FScript : Language Support for Navigation and Reliable Reconfiguration of Fractal Architectures.* Annals of Telecommunications Journal, Special Issue on Software Components - The Fractal Initiative, 2008 (to appear). This work is partially funded by SELFMAN.

- Boris Mejias and Peter Van Roy. *The Relaxed-Ring: a Fault-Tolerant Topology for Structured Overlay Networks*. Submitted to the Journal of Parallel Processing Letters. Accepted with revisions. First revision submitted, waiting for further comments by reviewers.

6.3 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*. 1st IEEE International Scalable Computing Challenge (SCALE 2008), Lyon, France. First prize award winning application and paper, May 2008. The Distributed Wiki is completely developed in SELFMAN.

- France Télécom R & D (Thierry Coupaye *et al*). The CLIF load testing framework won the Lutèce d'Or 2007 award for the best open source project made by a big company during the *Paris, capitale du libre* event organized by the Mairie de Paris and the French *Fédération Nationale de l'Industrie du Logiciel Libre*. The CLIF framework is partly developed in SELFMAN.

6.4 Dissertations

- Raphaël Collet. *The Limits of Network Transparency in a Distributed Programming Language*. Ph.D. dissertation, Department of Computing Science and Engineering, Université catholique de Louvain, Dec. 2007. Thesis jury: Marc Lobelle (UCL), Yves Deville (UCL), Per Brand (SICS), Joe Armstrong (Ericsson), Peter Van Roy (UCL). See Appendix of Deliverables document.

- Christophe Taton. *Toward self-optimization in autonomic systems* (in French). Thesis jury: Jacques Mossiëre, Sara Bouchenak, Peter Van Roy. Work in progress with Jean-Bernard Stefani at INRIA: infrastructure for self-configuration written in Mozart. Projected completion: Oct. 2008.

- Sylvain Sicard. *Self-repair in autonomic systems* (in French). Work in progress with Jean-Bernard Stefani at INRIA. Projected completion: Dec. 2008.

6.5 Book chapter

- Tallat M. Shafaat, Ali Ghodsi, and Seif Haridi. *Managing Network Partitions in Structured P2P Networks*. Book title: *Handbook of Peer-to-Peer Networking*. Springer. Chapter proposal accepted, full chapter submission due in August 2008.

6.6 Conference papers

- Peter Van Roy, Seif Haridi, Alexander Reinefeld, Jean-Bernard Stefani, Roland Yap, and Thierry Coupaye. *Self Management for Large-Scale Distributed Systems: An Overview of the SELFMAN Project*. Revised postproceedings of FMCO 2007, Amsterdam, The Netherlands, Oct. 2007 (to appear).

- Peter Van Roy. *Overcoming Software Fragility with Interacting Feedback Loops and Reversible Phase Transitions*. International Academic Research Conference on Visions of Computer Science, organized by the British Computer Society, Sept. 22-24, 2008 (to appear).

- Michael Lienhardt, Alan Schmitt, and Jean-Bernard Stefani. *Typing communicating component assemblies*. In 7th ACM International Conference on Generative Programming and Component Engineering (GPCE '08). This work is partially funded by SELFMAN.

- Sylvain Sicard, Fabienne Boyer, and Noel De Palma. *Using components for architecture-based management: the self-repair case*. In 30th International Conference on Software Engineering (ICSE 2008).

- Tallat M. Shafaat, Monika Moser, Thorsten Schütt, Alexander Reinefeld, Ali Ghodsi, and Seif Haridi. *Key-Based Consistency and Availability in Structured Overlay Networks*. In Proceedings of the Third International ICST Conference on Scalable Information Systems (InfoScale 2008), June 2008, Italy.

- P.-C. David, M. Léger, H. Grall, T. Ledoux, and T. Coupaye. *A Multi-staged Approach to Enable Reliable Dynamic Reconfiguration of Component-Based Systems*. 8th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'08), LNSC 5053, 2008 (to appear).

- Donatien Grolaux, Boris Mejias, and Peter Van Roy. *PEPINO: PEer-to-Peer network INspectOr*. Extended abstract for demonstrator. In

Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing (P2P'07). September 2-7, 2007, Galway, Ireland.

- Derrick Kondo, Artur Andrzejak, and David P. Anderson. *On Correlated Availability in Internet-Distributed Systems.* In 9th IEEE/ACM International Conference on Grid Computing (Grid 2008). Tsukuba, Japan, September 29-October 1, 2008.

- Felix Halim, Rajiv Ramnath, Sufatrio, Yongzheng Wu, and Roland H.C. Yap, *A Lightweight Binary Authentication System for Windows.* In IFIPTM 2008: Joint iTrust and PST Conferences on Privacy, Trust Management and Security, Trondheim, Norway, June 18-20, 2008.

- Felix Halim, Yongzheng Wu, and Roland H.C. Yap. *Security Issues in Small World Network Routing.* SASO 2008: Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems, Oct. 20-24, 2008, to appear.

- Stefan Plantikow, Alexander Reinefeld, and Florian Schintke. *Transactions and Concurrency Control for Peer-to-Peer Wikis: An evaluation.* Special CoreGrid volume, Springer Verlag, 2008.

- Monika Moser and Seif Haridi. *Atomic Commitment in Transactional DHTs.* 1st CoreGRID Symposium, Rennes, France, August, 2007.

- Thorsten Schütt, Florian Schintke, and Alexander Reinefeld. *A Structured Overlay for Multi-dimensional Range Queries.* In Euro-Par 2007, Parallel Processing, 13th International Euro-Par Conference. Rennes, France, Springer LNCS 4641, pp. 503 - 513, August 2007.

- Stefan Plantikow, Alexander Reinefeld, and Florian Schintke. *Transactions for Distributed Wikis on Structured Overlays.* In 18th IFIP/IEEE Distributed Systems: Operations and Management (DSOM 2007), San Jos, CA, USA, October 29-31, 2007.

- Tallat M. Shafaat, Ali Ghodsi, and Seif Haridi. *Handling Network Partitions and Mergers in Structured Overlay Networks.* In Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing (P2P'07). September 2-7, 2007, Galway, Ireland.

- Boris Mejias and Peter Van Roy. *A Relaxed-Ring for Self-Organising and Fault-Tolerant Peer-to-Peer Networks.* XXVI International Conference of the Chilean Computer Science Society (SCCC 2007), Nov. 8-9, 2007.

6.7 Conference submissions

- Boris Mejias and Peter Van Roy. *The Relaxed-Ring: a Fault-Tolerant Topology for Structured Overlay Networks.* Submitted to Eighth IEEE International Conference on Peer-to-Peer Computing.
- Boris Mejias and Peter Van Roy. *Partitioning and Merging the Ring.* Extended abstract for demonstrator. Submitted to Eighth IEEE International Conference on Peer-to-Peer Computing.
- Boris Mejias, Mikael Hogqvist and Peter Van Roy. *Visualizing Transactional Algorithms for DHTs.* Extended abstract for demonstrator. Submitted to Eighth IEEE International Conference on Peer-to-Peer Computing.

6.8 Workshop papers

- N. Jayaprakash, T. Coupaye, C. Collet, and P.-C. David. *Flexible Reactive Capabilities in Component-Based Autonomic Systems.* In 5th IEEE Workshop on Engineering of Autonomic and Autonomous Systems (EASe 2008), March 31st - April 4th, 2008, Belfast, Northern Ireland. This work is partially funded by SELFMAN.
- Boris Mejias, Donatien Grolaux, and Peter Van Roy. *Improving the Peer-to-peer Ring for Building Fault-tolerant Grids.* CoreGRID Workshop on Grid Programming Model, Grid and P2P Systems Architecture, Grid Systems, Tools and Environments, June 12-13, 2007, Heraklion - Crete, Greece Published in CoreGRID series, volume 7, *Making Grids Work.*
- Artur Andrzejak, Derrick Kondo, and David P. Anderson. *Ensuring Collective Availability in Volatile Resource Pools via Forecasting.* In 19th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2008) (part of Manweek 2008), Samos Island, Greece, September 22-26, 2008.
- P. Costa, G. Pierre, A. Reinefeld, T. Schütt, and M. van Steen. *Sloppy Management of Structured P2P Services.* 3rd Workshop on Hot Topics in Autonomic Computing (HotAC), Chicago, June 2, 2008.
- Boris Mejias and Jorge Vallejos. *Implementing Self-Adaptability in Context-Aware Systems.* In 6th International Workshop on Multi-paradigm Programming with Object-Oriented Languages (MPOOL 2007), at ECOOP 2007, July 31, 2007.

- Boris Mejias, Donatien Grolaux, and Peter Van Roy. *Improving the Peer-to-Peer Ring for Building Fault-Tolerant Grids.* In CoreGRID Workshop on Grid Programming Model, Grid and P2P Systems Architecture, Grid Systems, Tools, and Environments, FORTH-ICS, Heraklion, Greece, June 12-13, 2007.

- Stefan Plantikow, Alexander Reinefeld, and Florian Schintke. *Distributed Wikis on Structured Overlays.* In CoreGrid Workshop on Grid Programming Models, Grid and P2P System Architecture, Grid Systems, Tools and Environments, FORTH-ICS, Heraklion, Greece, June 12-13, 2007.

- Tallat Shafaat, Monika Moser, Ali Ghodsi, Thorsten Schütt, Seif Haridi, Alexander Reinefeld. *On Consistency of Data in Structured Overlay Networks.* CoreGRID Integration Workshop, Heraklion, Springer LNCS, April, 2008.

7 Article in eStrategies Projects Magazine: A Self-Managing Peer-to-Peer Network

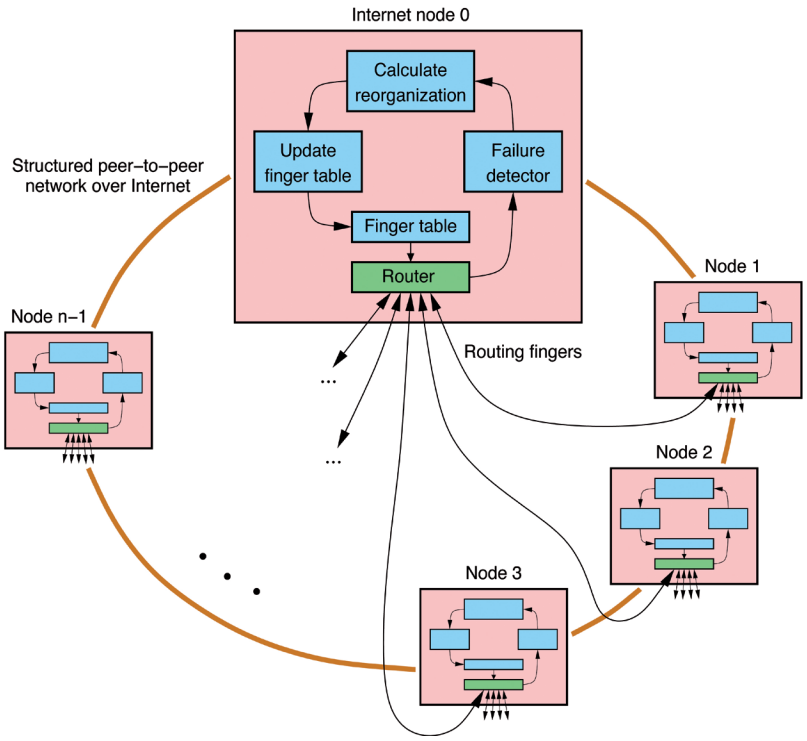
This section reproduces the three-page dissemination article that appeared in the magazine *eStrategies Projects* published by British Publishers in June 2008. Note that the final published article appears on pages 48-50 of the magazine (not 66-68 as in this preprint).

With distributed systems growing larger and more complex it is becoming increasingly clear that the task of managing them is beyond conventional technologies. Self-managing applications for the internet provide an effective, efficient and reliable solution, says **Peter Van Roy** of the SELFMAN project

A self-managing peer-to-peer network

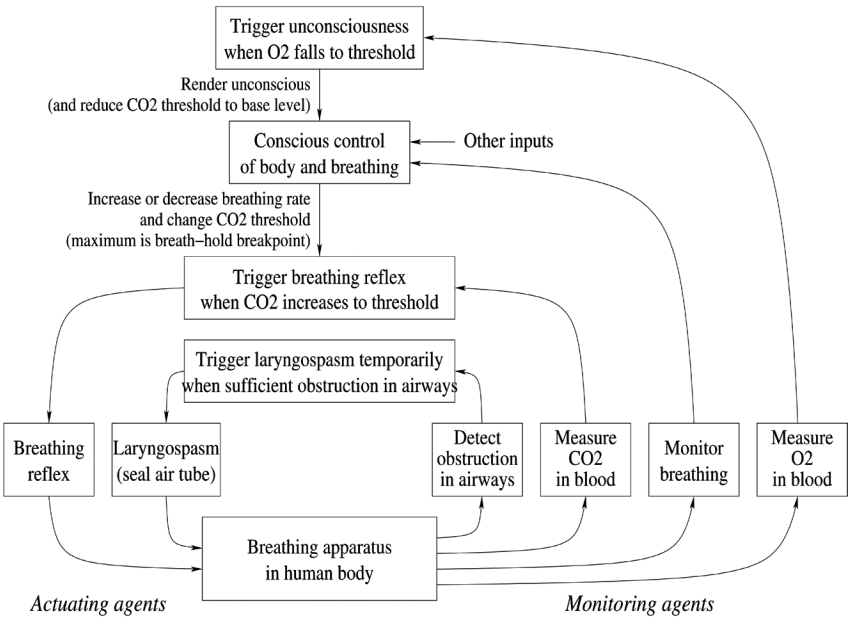
Over the past few years a number of new applications have appeared on the market that take advantage of the sheer scale of the internet: these include (among others) file-sharing, collaboration, social networks, role-playing games and vendors. However, ensuring that these applications are robust enough to meet the needs of users is a tough challenge, a challenge that nevertheless is being made ever more pressing by the inadequacies of current solutions. As things stand, keeping the kinds of applications alluded to above up and running requires the full attention of a team of specialists, and even then they often still have problems! Who amongst us has not seen a website crash because it has been 'slashdotted' or 'dugg?' Who has not seen applications break because of a missing plug-in or a network problem? These problems cause great frustration and no little inconvenience for users, who are often deprived of the ability to perform what can be essential work functions. The severity of the problem demands that it be addressed.

We at the SELFMAN project, an IST project launched in June 2006 to build self-managing applications on the internet based on peer-to-peer technology, aim to build internet applications that don't break and that don't need a team of specialists to keep them running. How can we do this? We take inspiration from the real world, from where the lesson is clear: things that take care of themselves are reliable and robust. This means that we need to make sure the applications are capable of self-managing. They should be able to reconfigure themselves so that they are capable of running as their users require regardless of the circumstances, even when their environment or their requirements change.



A self-managing peer-to-peer network (Figure 1)

In order to build self-managing applications the SELFMAN project starts with systems that have already solved the problem! These are the so-called structured overlay networks, which are the latest descendants of peer-to-peer technology. Peer-to-peer systems became popular with Napster (even though, with its centralised directory, it was not really a peer-to-peer technology), followed by systems like Gnutella, Kazaa, Morpheus, Freenet, and many others. In these systems all nodes are 'peers': they can all play the same roles. If a node crashes then the



Example of feedback loop architecture: Human respiratory system (Figure 2)

Feedback loops (Figure 2)

Self-managing systems react and adapt to changes in their environment. This behaviour is an example of a feedback loop that continuously monitors a parameter, calculates a response, and updates the system. A self-managing system contains many interacting feedback loops. For example, the figure shows the human respiratory system as four interacting feedback loops. This figure explains why choking is a normal defensive reaction (called 'laryngospasm') and why trained athletes can fall unconscious while holding their breath. A structured overlay network contains many interacting feedback loops. Within the SELFMAN project we are exploring how to program such a system to understand and predict its behaviour.

others take over and continue. Structured overlay networks are the latest and most advanced descendants of this idea: they provide guaranteed, efficient communication between their nodes as well as guaranteed lookup of data.

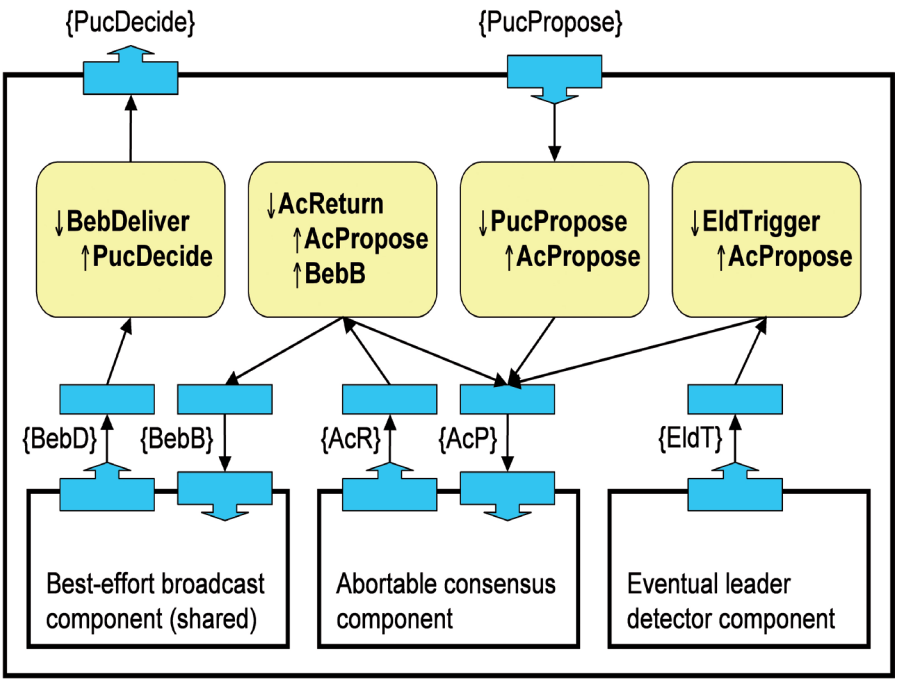
Structured overlay networks have been developed as a direct result of academic research. Our role at the SELFMAN project is to ensure that they can be of practical use in industrial applications. We are fixing the minor missing elements and are building a transaction service on top of them, while we have also made the overlay networks robust in the face of a critically important problem, namely network partitioning. When a network partitions – maybe, for example, because a router has gone down or a cable is cut – our networks continue to survive as separate overlay networks. When the connection returns, our separate networks merge together again automatically.

With these considerations firmly in mind SELFMAN is looking at three applications:

- A machine-to-machine messaging application (defined by France Telecom). This application creates a large ad-hoc network so as to enable the reliable transmission of messages across the world. If nodes go down or new nodes appear then the application has to keep working reliably and transparently.
- A distributed Wiki (as defined by the Zuse Institute in Berlin). Wikis are one of the world's most popular collaborative tools as they let a group of people create and organise large documents. However, when a Wiki has too many users performance suffers. We have built a Wiki over a structured overlay network using our transaction system for updates. This greatly improves performance and scalability. The distributed Wiki won first prize in the first IEEE International Scalable Computing Challenge (SCALE 2008).
- A video streaming application (defined by Peerialism AB). We want to distribute video on demand to large numbers of customers and to be able to guarantee quality of service on the internet. Customers come and go on a regular basis, sometimes they look at the same movies and sometimes they don't. In order to manage all these video streams we need to engage in dynamic reconfiguration. The video streaming application will soon be available as a product by Peerialism AB.

Crashes are particularly difficult on the internet because we cannot know for sure whether a node is really down or if it is just a communications problem. In response our solution uses a majority algorithm: we maintain several copies of the data, and the transaction can then commit if a majority of these copies are running

In order to build these applications we have implemented a transaction system on top of a structured overlay network. This is challenging because of the high rate of 'churn', that is, the high rate of nodes that leave, crash, or join, while the routing tables and storage



Example of software components: PAXOS uniform consensus algorithm (Figure 3)

self-organise to follow the churn. Crashes are particularly difficult on the internet because we cannot know for sure whether a node is really down or if it is just a communications problem. In response our solution uses a majority algorithm: we maintain several copies of the data, and the transaction can then commit if a majority of these copies are running. If the transaction manager node is suspected of failing then the algorithm picks another one transparently. This works even if the manager node has not really failed. This algorithm is based on a modified version of the PAXOS uniform consensus algorithm (Figure 3).

The SELFMAN project is entering its third and final year and we are committed to building on the advances we have made. We have already built a structured overlay network that can survive network partitions and that will merge when the network comes back together, as well as a transaction algorithm capable of handling the hostile internet environment. In the last year of the project we will be building our target applications using the overlay network and its transaction algorithm. We will also make our software available through open source and other licensing agreements, a key step towards entering a new era of robust, self-managing internet applications. ★

Software components (Figure 3)

All distributed systems, and especially self-managing systems, consist of many different parts that react in complex ways. For example, there are communication protocols (TCP, broadcast), gossip protocols, consensus protocols, and failure detectors. In order to make programming all these protocols and their interactions as easy and painless as possible, we program them as components. For example, the figure shows a uniform consensus component that uses a best-effort broadcast component, an abortable consensus component, and an eventual leader detection component. We at the SELFMAN project have defined a component model and implemented it in Java. All the distributed protocols are components that react to events. They are concurrent, compositional, and dynamically reconfigurable. As a bonus, the model is able to exploit multi-core architectures with no extra effort. The model is used by Peerialism for its video streaming application.

At a glance

Full Project Title
SELFMAN: Self Management for Large-Scale Distributed Systems based on Structured Overlay Networks and Components
European sixth framework programme, IST Research in Software Technologies

Project Partners
Université catholique de Louvain (UCL), Belgium
Royal Institute of Technology (KTH), Sweden
Institut National de Recherche en Informatique et Automatique (INRIA), France
France Télécom Research and Development, France
Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Germany
Peerialism AB, Sweden
National University of Singapore (NUS), Singapore

Contact Details
Peter Van Roy,
Université catholique de Louvain,
2 Place Sainte Barbe,
B-1348 Louvain-La-Neuve, Belgium
E: peter.vanroy@uclouvain.be
T: +32 485 42 46 77

Further Information
www.ist-selfman.org
cordis.europa.eu/ist/st/

Peter Van Roy

Project Coordinator
Peter Van Roy

Peter Van Roy is a professor in the Department of Computing Science and Engineering at the Université catholique de Louvain. Current research interests include self-managing systems and programming paradigms.

THE ADVENTURES OF SELFMAN

References

[1] Raphaël Collet. *The Limits of Network Transparency in a Distributed Programming Language*. PhD thesis, Dept. of Computing Science and Engineering, Université catholique de Louvain, December 2007.

[2] Rachid Guerraoui and Luís Rodrigues. *Introduction to Reliable Distributed Programming*. Springer-Verlag, 2006.

[3] Stefan Plantikow, Alexander Reinefeld, and Florian Schintke. Transactions for distributed wikis on structured overlays. In Alexander Clemm, Lisandro Zambenedetti Granville, and Rolf Stadler, editors, *DSOM*, volume 4785 of *Lecture Notes in Computer Science*, pages 256–267. Springer, 2007.