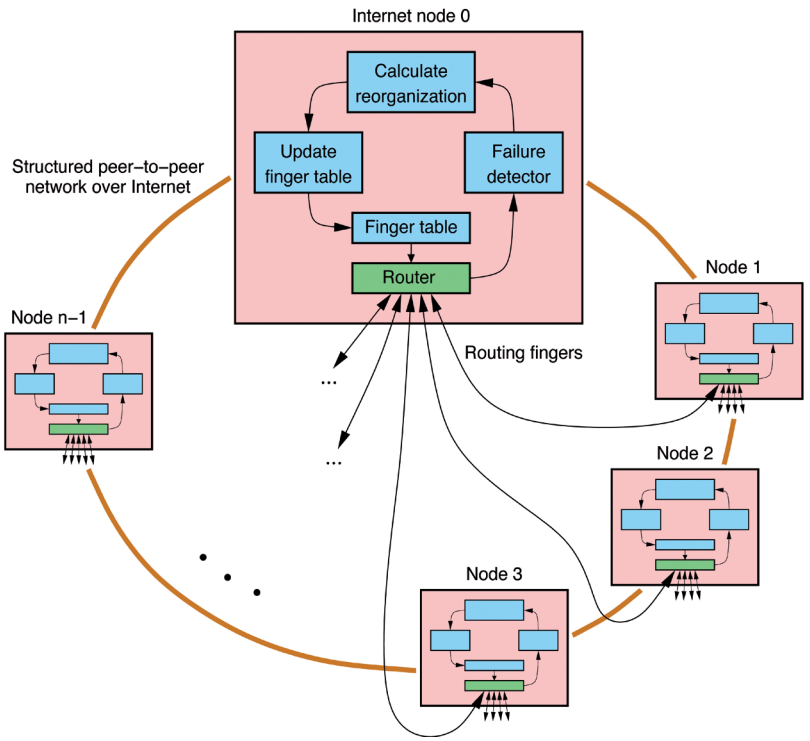
With distributed systems growing larger and more complex it is becoming increasingly clear that the task of managing them is beyond conventional technologies. Self-managing applications for the internet provide an effective, efficient and reliable solution, says **Peter Van Roy** of the SELFMAN project

A self-managing peer-to-peer network

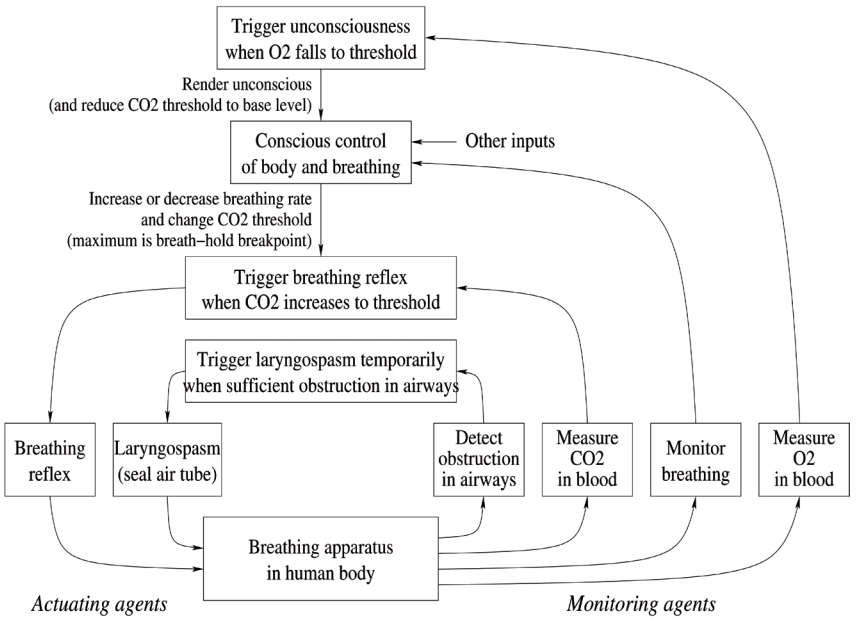
Over the past few years a number of new applications have appeared on the market that take advantage of the sheer scale of the internet: these include (among others) file-sharing, collaboration, social networks, role-playing games and vendors. However, ensuring that these applications are robust enough 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.

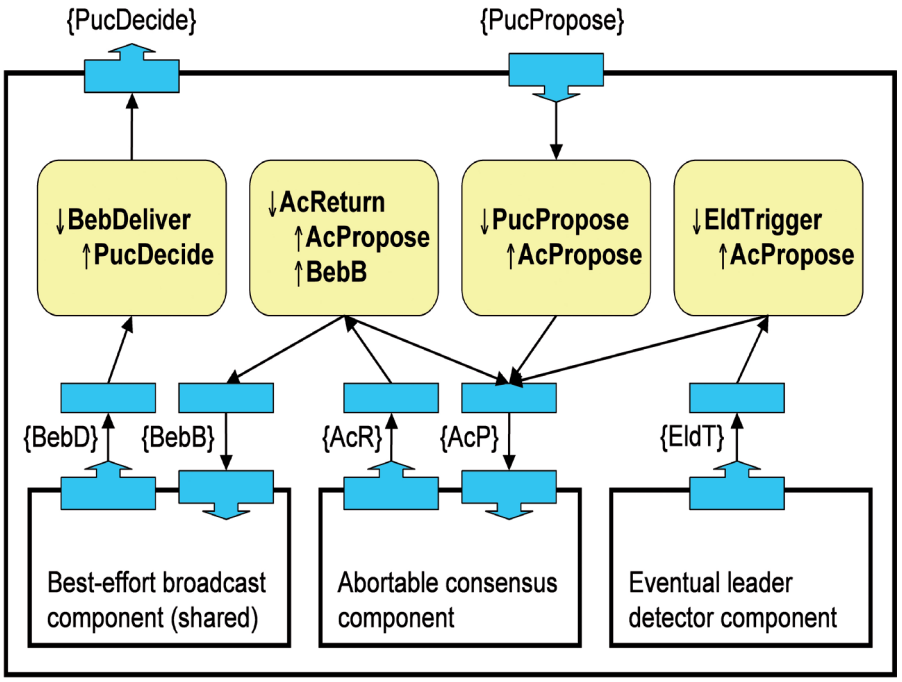
Structured overlay networks have been developed as a direct result of academic research. Our role at the SELFMAN project is to ensure that they can be of practical use in industrial applications. We are fixing the minor missing elements and are building a transaction service on top of them, while we 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