

Neighbour Discovery in Active Networks

Arso Savanovič¹, Borka Jerman Blažič¹, Janez Bešter²

¹Jožef Stefan Institute, Laboratory for Open Systems and Networks, Jamova 39, 1000 Ljubljana, Slovenia

²University of Ljubljana, Faculty of Electrical Engineering, Tržaška 25, 1000 Ljubljana, Slovenia

E-mail: {arso, borka}@e5.ijs.si, besterj@fe.uni-lj.si

Abstract. Active networks consist of traditional routers and active network nodes (ANN). Some aspects of active networks operation, e.g. security and interoperability, require each ANN to know its neighbour ANNs. However, neighbour ANNs are not directly connected, i.e. there may be one or more plain routers between them. This makes the discovery of neighbour ANNs a non trivial task and fundamentally different from physical neighbour discovery. In this paper we present a network protocol for automatic discovery of neighbour ANNs.

Key words: neighbour discovery, active networks, network protocol, self-organising networks

Odkrivanje sosednjih aktivnih vozlišč v aktivnih omrežjih

Povzetek. Aktivna omrežja so sestavljena iz navadnih usmerjevalnikov in aktivnih vozlišč (AV). Zaradi nekaterih vidikov delovanja aktivnih omrežij, npr. varnosti in medsebojne obratovnosti, mora vsako AV poznati sosedna AV. Vendar pa sosedni aktivni vozlišči na splošno nista neposredno povezani, t.j. med njima je lahko eden ali več usmerjevalnikov. Posledično je odkrivanje sosednjih AV netrivialno in se bistveno razlikuje od odkrivanja sosednjih usmerjevalnikov. V tem prispevku predstavljamo omrežni protokol za samodejno odkrivanje sosednjih AV.

Ključne besede: odkrivanje sosedov, aktivna omrežja, omrežni protokol, samoorganizirajoča omrežja

1 Introduction

The main goal of active networks is to enable fast, simple, and cost-effective deployment of new functionality in production routers and networks. This goal is facilitated with advanced routers based on a novel architecture, which enables installation and execution of programs within routers [16, 18]. These advanced routers with programming facilities are known as *active network nodes* (ANN). Each active node runs a Node Operating System (NodeOS) and one or more Execution Environments (EE) [5].

NodeOS is responsible for controlling the use of node's resources by other components. An EE implements a virtual machine which processes the packets and interprets active code. Each EE defines a different model of router programmability and these models are not interoperable. The architecture of an active node allows for multiple EEs to coexist in a single active node. Finally,

each EE can run a multitude of active codes, which provide novel and/or user specific end-to-end network services for the users and their applications.

In general, active networks consist of passive nodes (routers) and ANNs. ANNs distinguish themselves from traditional routers in that they provide additional functionality, which is not present in routers. Thus, a collection of ANNs in an active network form an overlay network of its own, which we call an *active overlay*.

An active overlay acts as a distributed system, so ANNs must operate in an organised and coordinated manner. However, the underlying IP infrastructure is highly dynamic, so it is desirable that active overlay be capable of self-organising and adapting to these changes, with minimum or no need for manual (re)configuration and maintenance by network operators.

Discovery of neighbour ANNs is one aspect of organised operation of ANNs. For example, the work on active networks security indicates that per-hop encryption may be required for protecting the integrity of active packets and active code [12, 15]. Per-hop encryption requires an ANN to establish a secret key with each of its neighbours. Consequently, an ANN needs to discover its neighbours to be able to run standard key-exchange protocols with them. In addition, some applications may require an ANN to consider EE availability when routing and forwarding packets. Thus, an ANN has to keep several lists of neighbour ANNs, one per each EE type. Then the EE specific routing and forwarding can consult the information in the neighbour list for the respective EE.

In an active overlay neighbours are generally not directly connected, i.e. there may be one or more plain routers between them. This makes neighbour discovery in

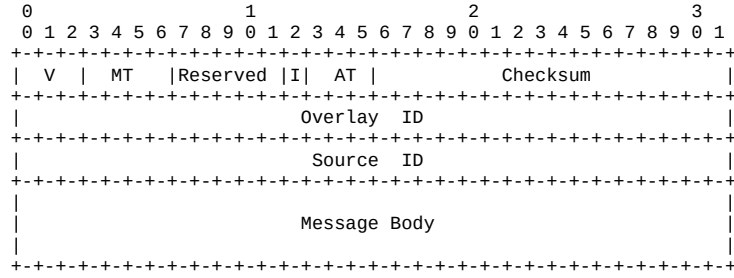


Figure 1. General protocol message format

an active overlay a non-trivial problem and fundamentally different from physical neighbour discovery. The neighbour relation between a pair of ANNs in an active overlay is defined by the underlying IP network. Two nodes A1 and A2 are neighbours in an active overlay if IP routes are defined such that there are no other ANNs on the path that packets traverse between A1 and A2.

2 A protocol for neighbour discovery in active overlays

We have developed a protocol for automatic discovery of neighbour ANNs [14]. In this section we give a short overview of the protocol.

The protocol has three operation modes, which run in parallel: node advertisements, discovery handshake, and heartbeat exchange. The protocol uses four messages. The general format of a protocol message, which is common across all messages, is depicted in figure 1.

The protocol message comprises a header and a body. The message header is common to all protocol messages, while messages differ by the format and semantics of the message body. Due to space constraints, we only describe the most important header fields.

Message Type (MT, 4 bits): Specifies the type of the protocol message.

Intermediary (I, 1 bit): The main purpose of this flag is to trigger the discovery handshake, if set in the Heartbeat message.

Overlay ID (32/128 bits): An identifier of the active overlay.

Source ID (32/128 bits): Identifier of the source node (see next paragraph).

A protocol message can contain one or more node IDs. A node ID is one of the addresses assigned to the node. A node can have several network interfaces, each with a different address assigned to it. However, nodes use a single, globally unique ID to identify themselves to other nodes. For example, much like in the OSPF routing protocol [11], a node can select its ID to be the lowest or the highest among its IP-addresses.

The format of all node ID fields in the message is determined by the value of the Address Type (AT) field in the header. Currently, the message can carry either a 32 bit IPv4 address or a 128 bit IPv6 address, which is defined by the value of the header field AT.

2.1 Node advertisements

An active node periodically sends the advertisement to an emulated communications bus, which then delivers the advertisement to all other ANNs connected to the bus. There is one emulated bus per active overlay.

The implementation of emulated communications bus is based on multicast [9]. Each active overlay uses one multicast group for coordination between ANNs. All members of the specific active overlay must join the respective multicast group. Where multicast is unavailable, it can be augmented with server based message distribution to members of an active overlay.

Active nodes advertise themselves with the Hello_World protocol message, which is identified by the value $MT = 1$ and has an empty message body.

When the Hello_World message arrives, the receiving node X only starts a discovery handshake with sending node Y, if it does not already keep state for that node, i.e. keep it in neighbour list or keep state on a running handshake with Y. In this case the receiving node starts the handshake with probability p , which is used to address the problem of *handshake synchronization*: if there are N members of the active overlay, then on average only $p * N$ nodes start a discovery handshake with sender Y simultaneously. The timeout value TO_{HW} defines the length of the period between the transmission of two successive messages Hello_World by a node.

2.2 Discovery handshake

The core of the protocol involves an exchange of two messages between the node which initiates the handshake (poller) and the node which responds (responder). The Hello message is sent by the poller and the responder returns the Hi message in response. Both messages are processed and modified by every active node en route between the poller and the responder.

The Hello message is identified by the value $MT = 2$. The body of the message contains two additional fields: the Destination ID and the Hop ID. Destination ID is the identifier (address) of the responder. Hop ID is the identifier of the current hop active node en route between the initiator and responder.

Upon reception of the Hello message the node first verifies the checksum and the overlay ID. Then it adds the previous hop ANN (Hop ID) to the neighbour list, if it is not already there. Finally, the node replaces the Hop ID value with its own identifier and forwards the Hello message to its destination.

The Hi message has the same format, except for the value of $MT = 3$. A node only transmits the Hi message as a response to the poller's Hello message. The Hi message is processed in a very similar manner as the Hello message.

The discovery handshake uses timeouts, retransmissions, and implicit acknowledgements to provide reliable message delivery service best suited to its needs. Use of TCP for this purpose is inappropriate for two reasons: its overhead is unacceptably high and its reliability service is not well suited for discovery handshake.

Reliable delivery is controlled by the poller according to the algorithm in figure 2. It uses the timer T_H and the timeout value TO_H to detect message loss. It also uses the retransmit counter N_H and the retransmit limit N_{H_MAX} to guarantee convergence.

Algorithm 1

```

1.  $N_H \leftarrow 0$ 
2. while  $N_H < N_{H\_MAX}$ 
3.   do reset timer  $T_H$ 
4.   send Hello message
5.   if Hi message received before  $T_H$  times out
6.     then stop
7.     else increment  $N_H$ 

```

Figure 2. Control of handshake reliability by the poller

Note that situations can arise, where the poller receives the Hi message after it has concluded the discovery handshake. The poller must silently drop an unexpected Hi message.

2.3 Heartbeat exchange

A node periodically sends the Heartbeat message to each of its known virtual neighbours to indicate that it is still alive. The Heartbeat message is identified by the value $MT = 4$. The body of the message only contains the address of the destination node (Destination ID). This message is also processed by all active nodes en route to destination. It is used for detecting changes in the network and for triggering the discovery handshake in this case.

A node knows that it is intermediary, if its ID is not equal to the Destination ID in the Heartbeat message. If two nodes exchange Heartbeats, then they are neighbours and there should be no intermediate ANNs between them. Therefore every intermediate ANN sets the I flag in Heartbeat, thus indicating that the network change occurred and that the receiver should start a discovery handshake.

The receiving node uses the Heartbeat message to refresh the state on the sending neighbour in its neighbour list and to trigger a discovery handshake with the sending node, if necessary. It uses the timer T_{HB} and the corresponding timeout value TO_{HB} for aging an entry in the neighbour list. It also uses the counter of consecutive missing Heartbeats from the source N_{HB} and the maximum limit on this count N_{HB_MAX} to guarantee convergence. The receiver first verifies the checksum and overlay ID and then processes the Heartbeat as depicted in figure 3.

Algorithm 2

```

1.  $N_{HB} \leftarrow 0$ 
2. while  $N_{HB} < N_{HB\_MAX}$ 
3.   do reset timer  $T_{HB}$ 
4.   if heartbeat received before  $T_{HB}$  times out
5.     then reset counter  $N_{HB}$ 
6.     get the value of the intermediary flag
       I from the message header
7.     get the source ID from the message
       body
8.     if the I flag is set
9.       then remove entry for source ID
           from the neighbour list
           start discovery handshake with
           source
10.    stop
11.    else increment counter  $N_{HB}$ 
12.    else increment counter  $N_{HB}$ 
13.    remove entry from the neighbour list

```

Figure 3. Processing of the Heartbeat message by the receiver

Note that situations can arise, where a node X receives the Heartbeat from some unknown node Y. Whenever such situation occurs at node X, it must initiate a new discovery handshake with node Y by sending the Hello message to Y.

3 Prototype implementation and experimental results

We have developed a proof-of-concept implementation of the protocol with the ANTS toolkit [19]. Each protocol message has been implemented in the form of a distinct ANTS capsule with simple forwarding routine. The core of the protocol has been implemented as a separate module, which operates on top of the ANTS system and pro-

vides features for neighbour discovery in an active overlay.

With the prototype, we were able to demonstrate the validity of the principles behind the discovery protocol, which were presented in this paper. Additionally, we have done some initial experiments with the protocol prototype. We were primarily interested in the impact of protocol parameters on protocol convergence delay. We have observed the operation of the protocol in a controlled environment of a single branch of an active overlay, where we introduced basic types of network perturbations in a controlled manner and measured the convergence delay. We tried to experimentally estimate the upper bounds on the neighbour discovery protocol convergence delay, given the values of protocol parameters.

It is sufficient to observe only two basic types of network perturbations, which we call representative perturbation types: addition of a new ANN and removal of an existing ANN. Other types of basic network perturbations are generally equal to the combination of these two.

When an ANN is removed from an active overlay, the upper bound on the convergence delay is given with (1). The protocol converges when node X deletes Y from its neighbour list. The neighbour discovery protocol always converges, when a node is removed from an active overlay, therefore (1) is a *deterministic* upper bound on convergence delay. As an example, the typical values of $TO_{HB} = 10s$ and $N_{HB_MAX} = 3$ yield the deterministic upper bound on convergence delay at ANN removal of $T_{DC_MAX} = 30s$.

$$T_{DC_MAX} = N_{HB_MAX} * TO_{HB} \quad (1)$$

$$T_{UC_MAX} = N * TO_{HW} + N_{H_MAX} * TO_H \quad (2)$$

However, due to probabilistic aspects of its operation, the neighbour discovery protocol does not necessarily converge, when a new ANN is added to an active overlay. Therefore we talk of *probabilistic* upper bound on protocol convergence delay in this case. The probabilistic upper bound is defined such, that (a) a sufficiently high convergence probability is chosen, e.g. 99%, which is considered as an event of protocol convergence, (b) from experimental results, the required number N of periods TO_{HW} is obtained, which yields the given convergence probability, and (c) given N and other protocol parameters, the upper bound on convergence delay is then obtained from (2). The first factor in (2) represents the delay till the start of the successful handshake, while the second factor represents the maximum time required for the successful handshake to complete.

In every experiment we have fixed the protocol parameter values and repetitively measured the convergence delay $T_{UC_MAX}(N)$. We then used (2) to calculate the

corresponding value of N for each data sample. Thus, relative frequencies $f_C(N)$ of different values of N have been obtained for every experiment. In a large number of experiment repetitions these relative frequencies converge towards the protocol convergence probability $P_C(N)$.

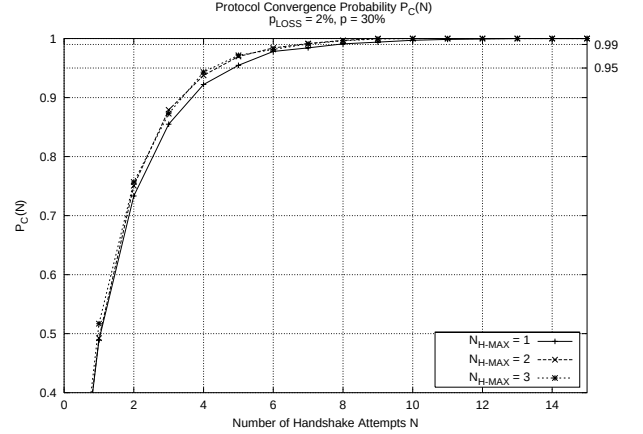


Figure 4. Impact of parameter N_{H_MAX} on protocol convergence probability, when a new ANN is added

The impact of parameter N_{H_MAX} on protocol convergence probability is depicted in figure 4. With typical losses in the network ($p_{LOSS} = 2\%$), convergence probability is practically not affected by N_{H_MAX} , as indicated by the results. However, it is still beneficial to allow at least one retransmission of the Hello message ($N_{H_MAX} = 2$) to make the protocol more robust to occasional periods of congestion, when loss probability in the network increases temporarily.

To illustrate the typical values of upper bound on protocol convergence delay when a new ANN is added, consider the following. In case that convergence probability of 99% is required, the results in figure 4 suggest that $N = 7$ periods TO_{HW} are required for the protocol to converge with this probability. Given this value of N and typical values of $TO_{HW} = 5s$, $TO_H = 1s$, (2) yields the probabilistic upper bound on protocol convergence delay of $T_{UC_MAX} = 37s$. The value of TO_H must be higher than the maximum round trip time in the network (typically less than $1s$), while the value of TO_{HW} has been chosen based on the target convergence delay value.

As seen above, the choice of parameter values was driven by our goal to achieve the convergence delay comparable to that of current implementations of the OSPF routing protocol, which is typically around 30s [17, 2]. We find the comparison with OSPF interesting due to the following: (a) as far as the authors are aware, no results on convergence delay are currently available for related efforts in the area of automatic discovery of neighbour ANNs (e.g. [10]), (b) routing protocols provide analogous functionality to neighbour discovery protocol: after a perturbation occurs in the network, the routing proto-

col has to accordingly reconfigure the physical network (i.e. routes in the network) as soon as possible in order to minimise the disruption in communication caused by the perturbation, and (c) OSPF is among the fastest routing protocols, i.e. among the protocols with the smallest convergence delay.

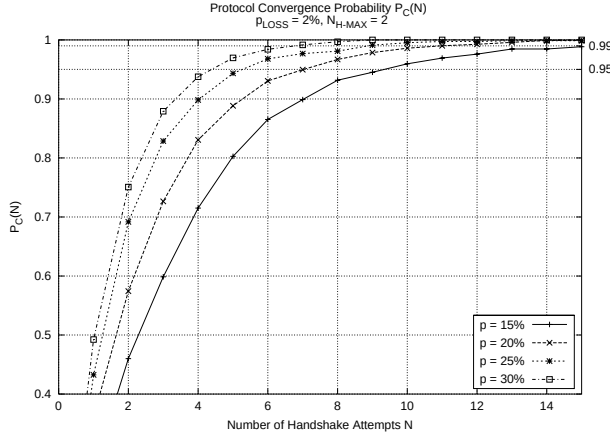


Figure 5. Impact of the handshake probability p on protocol convergence probability, when a new ANN is added

Figure 5 depicts that the handshake probability p significantly impacts the protocol convergence probability when a new ANN is added. For example, when $p = 30\%$, only $N = 7$ periods of TO_{HW} are required for the convergence probability of 99%. When $p = 15\%$, $N = 16$ periods are required for the same convergence probability. Given that the first factor of (2) is predominant, the value of $p = 15\%$ more than doubles the convergence delay in comparison to the value of $p = 30\%$.

4 Related work

In existing networks, there are several neighbour discovery protocols, for example ICMP Router Discovery [6], IPv6 Neighbor Discovery [13], and OSPF Hello exchange [11]. However, these protocols are based on the assumption that neighbouring nodes are connected to the same physical link. These protocols do not work in active overlays, where two neighbour nodes are not necessarily directly physically connected.

Application Layer Active Networks (ALAN) [7] and Active Network Overlay Network (ANON) [4] explicitly address the notion of active overlays. ALAN is a specific model of active networking that facilitates programmability not by enhancing routers but rather by adding dedicated servers to the network for execution of active code. An integral part of ALAN is the discovery protocol, but this protocol is only aimed at building a large distributed database of all active nodes [8].

ANON is an experimental protocol, which has been defined by the active network community to enable in-

terconnection of EEs [4]. Similarly to ALAN discovery, core ANON nodes (called hubs) use the protocol message “announce_neighbours” only to maintain an updated distributed database of core ANON nodes.

The ABone is a virtual active network testbed, which has been initially established for the active network research program funded by DARPA [1]. Activities on design, construction and operation of the ABone have resulted in some novel technologies. One of these is the proposed ABone mechanism for dynamically extending the ABone topology called DANTE—Dynamic Active Nets Topology Extension [3]. DANTE enables dynamic addition of active nodes at the edge of the ABone. However, it is limited by the assumptions that the ABone core is static and that the attachment point (node) is statically pre-configured.

The RADAR protocol [10] is an extension of the ANTS toolkit, which enables ANNs to dynamically discover neighbour ANNs. This effort is closely related to our own work and the two bear many similarities. However, the two solutions differ in their main goals and technical approaches. RADAR does not search for all neighbour ANNs, since it is concerned with the protocol bandwidth consumption. Instead, it configures a sub-optimal active overlay, i.e. a “good” match to the underlying IP-infrastructure. This way, it strives to achieve a low protocol bandwidth overhead for the cost of configuring suboptimal routes between neighbour ANNs in an active overlay. Contrary to RADAR, our protocol strives to keep an active overlay the *optimum* match to the underlying IP-infrastructure at all times. Therefore, we are primarily concerned with the protocol convergence speed. We believe that the use of optimal routes between ANNs can yield bandwidth gains which outweigh the additional bandwidth overhead incurred by the neighbour discovery protocol.

5 Conclusions and future work

We have shown that (sets of) active nodes form active overlays. The core of this paper and its main contribution is the definition of the protocol, which enables active nodes to automatically discover their “direct” neighbours in those active overlays that a node is a member of. This eliminates the need for cumbersome manual (re)configuration of active nodes and contributes towards the development of self-organising active networks.

We have developed a proof-of-concept prototype of the protocol to verify and demonstrate the validity of its basic principles. We have also carried out several experiments with the prototype. Experimental results indicate that the protocol reacts to network perturbations with quite reasonable speed. The protocol as presented in this paper does not cover some special cases, such as directed IP-networks and load balancing over equivalent multiple

paths. However, the work is already underway to add extensions to the basic protocol that address these situations. Future work also includes formal analysis of the protocol and further experiments to gain a more thorough understanding of the protocol operation.

6 References

- [1] ABone. <http://www.isi.edu/abone/>.
- [2] A. Basu and J. G. Riecke. Stability Issues in OSPF Routing. In *Proceedings of SIGCOMM 2001*, pages 225–236. ACM Press, New York, 2001.
- [3] S. Berson and B. Braden. DANTE: Dynamic topology extension mechanism for the ABone. ABone Technical Specification, mar 2000.
- [4] B. Braden, M. Hicks, and C. F. Tschudin. Active Network Overlay Network (ANON). DARPA Active Networks Group Draft, dec 1997.
- [5] K. L. Calvert. Architectural Framework for Active Networks version 1.1. Technical report, University of Kentucky, dec 2001.
- [6] S. E. Deering. ICMP Router Discovery Messages. RFC 1256, sep 1991.
- [7] M. Fry and A. Ghosh. Application Level Active Networking. *Computer Networks*, 31(7):655–667, 1999.
- [8] A. Ghosh, M. Fry, and J. Crowcroft. An Architecture for Application Layer Routing. In *Proceedings of IWAN 2000*, pages 71–86, 2000.
- [9] D. Kosiur. *IP Multicasting*. John Wiley & Sons, Inc., 1998.
- [10] S. Martin and G. Leduc. Radar: Ring-based adaptive discovery of active neighbour routers. In *Proceedings of IWAN 2002*, pages 62–73. Springer, dec 2002.
- [11] J. Moy. OSPF Version 2. RFC 2328, apr 1998.
- [12] S. L. Murphy, E. T. Lewis, and N. M. R. Watson. Secure Active Network Prototypes. In Werner Bob, editor, *Proceedings of DANCE 2002*, pages 166–181. IEEE Computer Society, 2002.
- [13] T. Narten, E. Nordmark, and W. A. Simpson. Neighbor Discovery for IP version 6. RFC 2461, dec 1998.
- [14] A. Savanović. Automatic Discovery of Neighbour Active Network Nodes. Technical Report IJS DP-8725, Jožef Stefan Institute, jan 2003.
- [15] A. Savanović, D. Gabrijelčič, B. Jerman Blažič, and S. Karnouskos. An Active Networks Security Architecture. *Informatica*, vol. 26(2), pages 211–221, 2002.
- [16] J. M. Smith, K. L. Calvert, S. M. Murphy, H. K. Orman, and L. L. Peterson. Activating Networks: A Progress Report. *IEEE Computer Magazine*, pages 32–41, apr 1999.
- [17] W. R. Stevens. *TCP/IP Illustrated: The Protocols*, volume 1. Addison-Wesley, fourth edition, 1994.
- [18] D. L. Tennenhouse, J. M. Smith, W. . Sincoskie, D. J. Wetherall, and G. J. Minden. A Survey of Active Network Research. *IEEE Communications Magazine*, pages 80–86, jan 1997.
- [19] D. J. Wetherall, J. V. Guttag, and D. L. Tennenhouse. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. In *Proceedings of OpenArch 1998*, pages 117–129, apr 1998.

Arso Savanović received the B.Sc. and the M.Sc. degrees in Electrical Engineering from the University of Ljubljana, Ljubljana, Slovenia, in 1996 and 1999, respectively. Currently, he is working towards his Ph.D. He joined Jožef Stefan Institute in 1997. He has been involved in the international projects NICE, EXPERT, UNIVERSAL, and FAIN from the EU R&D Framework Programmes. His research interests include active networks, security, broadband networks, and multimedia communications.

Borka Jerman Blažič is Head of the Laboratory for Open Systems and Networks at Jožef Stefan Institute, Ljubljana, Slovenia. She is teaching courses on Telecommunication services at the Faculty for Economics, University of Ljubljana. She is a member of the ARNES (Slovene academic network) Technical Committee, member of TERENA Technical Committee and convenor of TERENA (Trans European Research and Academic Networks Association) WG on Internationalisation. She is chairing the national standardization committee on information technologies (JTC1) and is representing Slovenia in ISO JTC1 SC2, JTC1 SC22WG20 and CEN TC304. She is author of more than 200 published scientific papers, invited lectures on international conferences and workshops. She is currently coordinating the Slovenian part of three Fifth Framework projects, UNIVERSAL dealing with brokerage platform for learning resource exchange between higher education institutions, ELENA dealing with development and deployment of smart learning spaces based on heterogeneous services, and FAIN dealing with active networks technologies.

Janez Bešter graduated in 1979. He received his M.Sc. and Ph.D. degrees in the field of telecommunications from the University of Ljubljana, Slovenia, in 1982 and 1995, respectively. Currently, he is Head of the Laboratory for Telecommunications at the Faculty of Electrical Engineering in Ljubljana and Professor for subjects, such as Telecommunication Basics II, Computation Systems and Networks, Communication Networks and Services, Design, Modeling and Management of Telecommunication Networks and Intelligent Networks. Twenty years of his research and development activities, as well as teaching, have focused on the field of planning, realization and management of telecommunication systems and services, together with the application of information technologies in education. His current work involves research and engineering in the fields of IP, ATM, TM (Telemanagement), and IIN (Integrated Intelligent Networks). He is also a member of the IEEE and ACM as well as an associate member of the IEICE.