ARCHITECTURES FOR AUTOMATIC WAN TOPOLOGY INTERCONNECTION

Petr Grygárek, David Seidl
Department of Computer Science, Faculty of Electrical Engineering and Computer Science
Technical University of Ostrava, Tř. 17. listopadu, 708 33 Ostrava, Czech Republic
Tel.: (+420) 597 323 263, Fax: (+420) 597 323 099
E-mail: petr.grygarek@vsb.cz, http://www.cs.vsb.cz/grygarek

Abstract.
In practical education in networking laboratories it is useful to be able to interconnect various WAN topologies quickly and efficiently or automatize the interconnection based on demands of students who access the laboratory remotely. In the article we present our currently developed architectures for automatic interconnection of WAN topologies and hardware device prototypes we constructed for that purpose. The presented technologies also support tunnelling of WAN traffic between multiple distant sites so that they can be used to create distributed virtual WAN topologies.

Keywords: Communication Technologies, Virtual Laboratory

1. INTRODUCTION

During practical education in networking laboratory, it is necessary to work on various topologies of networking devices. Unfortunately, the process of connecting network topology is both time-consuming and error-prone and commonly prevents students to concentrate on the configuration of particular protocol or technology, which is the main objective of the respective lesson. Our experience revealed that although it is inevitable to let students connect network topologies manually at the beginning of their study to give them concrete idea about WAN/LAN interfaces usage, it is much more effective to concentrate on upper-layer protocols and don't waste time with repeated physical layer troubleshooting later. The another issue with frequent topology changes is that students often do not manipulate network interface connectors with enough care. This often results to mechanical damage of connecting cables or networking device module's connectors and a need of their replacement. This problem is most serious in cases of non-modular routers, where damaged interface cannot be easily replaced and with new tiny connector types, which are very frangible.

To avoid the above mentioned problems, we searched for methods how to interconnect network topologies automatically without human interaction. The result was the Distributed Virtual Crossconnect used in our Distributed Virtual Networking Laboratory [2] architecture, which may be configured as a single entity although it is composed of multiple switching elements of various types. For LAN (Ethernet) interconnection, we use standard LAN switches and VLAN-based interconnection. Using VLAN tunneling (also called dot1QinQ sometimes), we are even able to interconnect trunk links of various devices and have the switching element be completely invisible for the laboratory devices. Unfortunately, there exists no similar commercially-available and cheap solution for WAN links. This is why we decided to develop a series of our own devices for WAN port automatic interconnection. The basic ideas, architectures and experiences with these devices will be discussed in the following article.

2. THE FIRST GENERATION CROSSCONNECTS

Both crossconnect prototypes we developed up to now have the similar philosophy (fig. 1). All network devices' WAN ports are connected to interfaces of a single crossconnect which can be configured to interconnect arbitrary pair of connected ports. The configuration is accomplished via RS-232 console port using a simple "IOS-style" command line interface (CLI) available to instructor or lab administrator. As in IOS, command completion, possibility of use of abbreviations and context-based help system was implemented. The configuration is maintained in RAM but may be also stored into internal flash memory so that it can be loaded automatically when the crossconnect device is powered on. The device is controlled by Atmel 8051ED2 microprocessor which acts as a CLI command interpreter and configures switching array according to user's requirements.

In some cases, it is useful to be able to control the crossconnect not only by directly connected RS-232 terminal, but also remotely via intranet or Internet. Our solution for RS-232 to TCP conversion was to use relatively cheap commercially-available modules, in this case Charon II [5] for mutual RS232 to Ethernet conversion and Sollae EZL80c [4] for RS-232 to WiFi (802.11b) conversion, as can be seen at fig. 1.

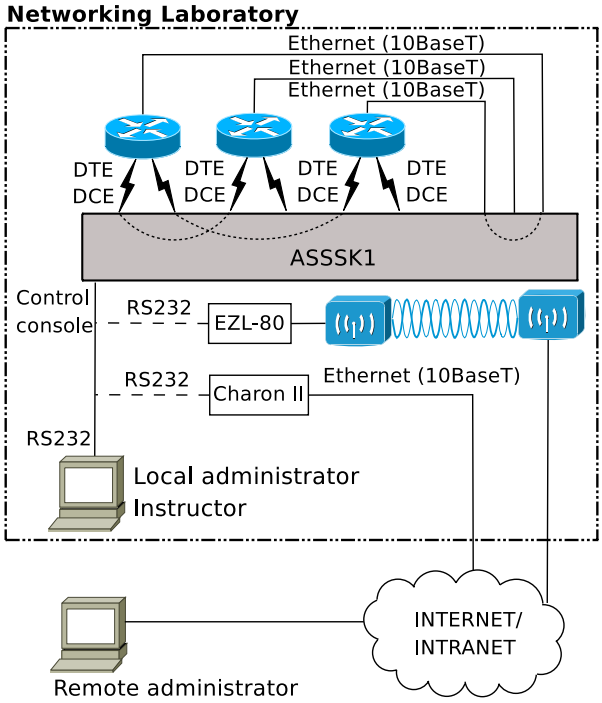

Fig. 1. Basic Crossconnect Philosophy

As will be described in detail later, we subsequently tested and implemented various approaches for actual signal switching, but the substantial part of controlling software was reused in the individual prototype devices.

2.1 WAN INTERFACE TYPE SELECTION

One of the main issues of crossconnect design was a selection of physical WAN interface type our device should support. Since we use our crossconect primarily to interconnect WAN ports of Cisco routers which can provide multiple physical interface and let the user to choose the required one only by usage of an appropriate WAN cable, we could choose from ITU-T X.21, ITU-T V.35, EIA/TIA RS-232 and EIA/TIA RS-499 (all synchronous). Since we wanted to limit a number of signals to switch and also take the interface's connector availability into account, we chose to use RS-232 because it doesn't use symmetrical signals neither for data nor control and it's CANNON DB-25 connector is cheap and easily available. In fact, we use "null modem" interconnection so that we need only to switch RxD and TxD signals and provide clock signals, as will be discussed later.

Another important issue we had to solve was the problem of clocking. In reality, routers at both sides of the leased line provided by telco act as DTEs and are connected with synchronous modems using „DTE cable". Clocking signal for both routers is provided by respective modems. In laboratory, it is inefficient to have so many modems, so in case of Cisco routers most people commonly connect their WAN interfaces directly, using a pair of two different cables for DCE and DTE side. The router connected with a "DCE" cable provides clocking for both directions of the communication if instructed to do so by an IOS command. It means that for that type of direct interconnection we are only able to connect a pair of interfaces if one interface is connected with a DCE cable and another one with a DTE cable. It means that we have a problem with cable type to use if we want to connect all WAN ports to the single crossconnect and be able to connect arbitrary pairs of connected ports.

To allow crossconnect to connect arbitrary pair of WAN ports, we decided to take completely different approach, which much more resembles the reality. All WAN ports are connected to the crossconnect using DTE cables and the crossconnect itself provides clocking for all devices, exactly as would a modem at the end of leased line do. In the newer prototype, we are even able to set various clockrates for individual pairs of connected ports. From the student's perspective, the crossconnect may be viewed as a telco cloud which provides leased line services including clocking and he/she does not have to take care about clocking at all.

2.2 THE ANALOG CROSSCONNECT

The very first version of the crossconnect (called ASSSK-1) was developed by David Seidl in his MSc. Thesis [1]. The general aim of the thesis assignment was to develop a crossconnect based on analog switch array core suitable for connecting of signals of various networking technologies. Individual ports are attached to the switching core via interface modules, which adapts various electrical interfaces' signals to the voltage range suitable for the switching core. The device (fig. 2) may accommodate up to 16 modules.

Fig. 2. ASSSK-1: The Analog Crossconnect

The switching core is composed of two Zarlink MT8816 analog switch array integrated circuits [6], which together from a 16 x 16 matrix. The matrix allows to connect each of 16 TxD signals to any of 16 RxD signals, so that arbitrary 8 pairs of modules may be connected together. It is even possible to loopback any interface for testing purposes.
Various interface modules may be developed to switch signals of individual interface types. The only limitation is the frequency range of the used switching array (30MHz).

That range proved sufficient for 10BaseT Ethernet (three harmonics are normally enough) and synchronous RS-232 up to 2 Mbps. We developed a double-interface modules, which allow to connect either RS-232 WAN port or 10BaseT Ethernet. Because of different electrical characteristics of RS-232 and Ethernet signals, it proved most effective to use a small mechanical relay to choose which of the two interfaces available at the module will be really connected to the switching core. The reachable bitrate of switched serial WAN interfaces is somewhat limited by RS-232 to TTL convertors (the capacitor-based charge pump) used to create ±12V for RS-232 interface), but we currently plan to provide external 12V DC supply to avoid this problem.

The block diagram of the ASSSK-1 device is depicted at fig. 3. The Control Processor interacts with user using CLI and configures analog switching array. We decided to use separate microprocessor (the Clock Processor) to provide clocking for individual WAN lines.

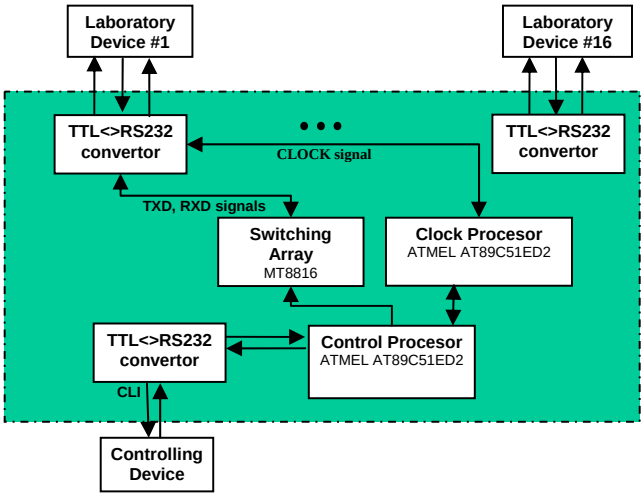

Fig. 3. ASSSK-1 Block Diagram

To support potential future extensions, we decided to provide I2C bus implemented by ATMEL microprocessor for the usage at the interface modules. It means that various I2C-enabled devices may be attached the similar way as interface modules and accessed from the control microprocessor. Currently we decide the attachment of external Flash memory to let instructors to store multiple pre-defined crossconection configurations and easily choose one of them. By implementing of a simple user interface consisting from a numeric keyboard and LCD display, we can also provide a mechanism to let the instructor change configurations very quickly and efficiently without a need of external control terminal (most probably PC).

2.3 FPGA-BASED CROSSCONNECT

After a period of usage of ASSSK-1, we decided to redesign an architecture based on the experience with the original prototype. Switching of Ethernet ports proved inefficient, because we need to switch faster interfaces than 10BaseT, which is not possible because of frequency limitation of the used analog switch array. This is why we decided to concentrate on WAN interface switching in the future and interconnect Ethernet ports using standard VLAN-aware 10/100/1000 Ethernet switches and VLAN/802.1q tunneling approach, which is also cheaper.

The main aims of the new crossconnect version was to make the device more replicable and compact, decrease it's production cost and increase flexibility. It also proved unnecessary to be able to switch various WAN physical interface types, since RS-232 proved most efficient during usage period of the ASSSK-1. This is why we abandoned the modular architecture and decided to implement interface circuitry on the baseboard instead of at modules. We also used FPGA technology and VHDL to implement the device much more efficiently. The most important change is that the FPGA-based switching core is now fully digital. Except the switching function, the FPGA circuit also provides clocking for individual ports based on the frequencies pre-set to it's configuration registers by controlling microprocessor.

The prototype (fig. 4) was developed by Petr Sedlar in his master thesis [3], produced and successfully tested. Much higher bitrates are reachable with digital switching matrix than with the previous analog one. Twenty interfaces are available at the chassis. The device is very easy to modify because of it's capability of in-system reprogramming of Atmel control microprocessor and FPGA core.

Fig. 4. ASSSK-2

The block diagram of the FPGA-based device called ASSSK2 is depicted at fig. 5.

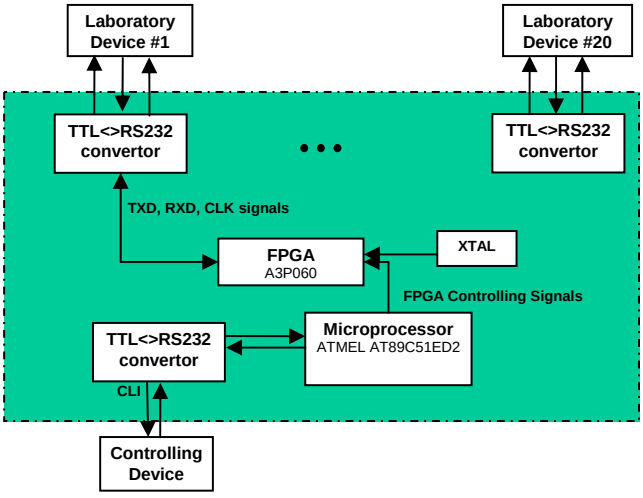

Fig. 5. ASSSK-2 Block Diagram

Because of implementation of interface circuitry on the baseboard instead on interface modules in ASSSK-2, the number of mechanical contacts was reduced considerably so the new crossconnect is not only cheaper and more compact, but also more reliable.
In the future, we intent to integrate the whole logic, including the control processor, into the more advanced type of FPGA integrated circuit. We expect that it will further decrease the cost and extend the flexibility of the crossconnect.

3. SECOND GENERATION CROSSCONNECTS

The limitation of first generation of our crossconnect solutions was that there was no possibility to switch traffic across multiple crossconnect instances. Although there is 16 or 20 ports at the available models, the scalability is somewhat limited because even if we produce multiple crossconnect instances, we are only able to connect WAN ports of network devices connected to the same crossconnect. This is why we searched for solutions how to let traffic pass across multiple crossconects connected together. We call crossconnects with this capability a second generation crossconnects.

The first idea was just to reserve some number of standard crossconnect ports for interconnections between crossconnect (called trunk ports in the following text). Unfortunately, this solution would be rather inefficient because 4 ports in total would have to be consumed for a single interconnection of two WAN ports between a pair of crossconnects. It would be even more in case of a longer chain of a crossconnects linked in this way. A non-trivial issue of clock synchronization between multiple crossconnect devices would have to be solved also. Although it is of course possible to connect crossconnects into more efficient hierarchical structures than a simple chain, we decided to take completely different approach.

The solution for passing traffic between multiple crossconnects we are working on now was influenced by a need to pass traffic between crossconnects located at various, physically distant sites [7]. The general idea is not just to interconnect physical signals, but read the content of passed PPP/HDLC frames, encapsulate them and tunnel over intranet or even Internet. Frames are decapsulated at the receiving side and sent out of the particular serial port to the WAN interface of the network device they are destined to. This approach scales well because the interconnections between individual crossconnects forms a logical full mesh and in case of sufficient capacity of the underlying LAN/WAN the number of interconnections of network devices connected to different crossconnects is potentially unlimited. We denote solutions adhering the approach described above as a second-generation crossconnects.

3.1. LINUX-BASED CROSSCONNECT

The simplest way to process HDLC/PPP frames and pass them between multiple switching devices is to use software-based approach. We decided to utilize PC with Linux for that purpose. The architecture we are now experimenting with is described at fig. 6. The PC is equipped with multiport synchronous serial card, which allows to connect WAN interfaces of network devices to be interconnected. We investigate now how standard Linux PPP/HDLC drivers could be modified to allow a switching software developed for that purpose to switch PPP/HDLC frames between logical ppp/hdlc interfaces. The switching software will also be able to tunnel frames between multiple crossconnect PCs in UDP datagrams, so that we will be able to create virtual WAN links over local Ethernet segment or over the Internet. The remote configuration using Telnet is planned in the first prototype.

Because the commercially available synchronous serial cards are both costly and typically do not provide more than two ports, we started to work on our own FPGA-based card with a higher port density (8 ports as a minimum). It is not completely clear now how quickly we will be able to switch frames between individual interfaces using standard PC, so we count with 64kbps bitrates on individual WAN ports at the beginning, which is completely sufficient for educational purposes. We denote the above mentioned Linux-based crossconnect architecture as ASSSK-3.

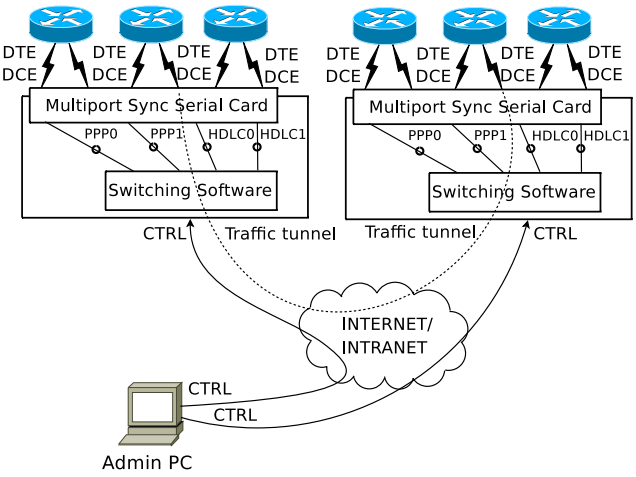

Fig. 6 –ASSSK3: The Linux-based Crossconnect

3.2. EXTENSION OF FPGA-BAASED CROSSCONNECT

Another possible approach we are assessing now for processing of PPP/HDLC frames and tunelling them across the Ethernet is to build-in a frame processing intelligence into the current FPGA-based crossconnect (ASSSK-2). The frame separation logic capable of recognizing frame flags and handling bit stuffing seems to be relatively simple to integrate into FPGA. We plan to use some commercially-available embedded module for synchronous serial to

Ethernet conversion in the prototype, such as above-mentioned Charon II module. The frame separation logic could be also implemented at this module. The general idea is to reserve a couple of ports of FPGA-based switch array and connect them internally to the serial-port side of the serial-to-Ethernet conversion modules. The way how the module will handle frames incoming from switch array and Ethernet interface will be programmed to the module by crossconnect's control processor. In fact, only destination/source MAC (or IP) address to internal switching array port mapping will have to be configured. The FPGA switch array will then switch frames coming from another crossconnects through Ethernet and serial-to-Ethernet module to it's internal port the same way as if it came from the regular port. Multiple serial-to-Ethernet convertors can be integrated together to limit the size and cost of the construction or a more-powerful convertor capable of handling multiple serial ports could be utilized. The architecture proposal is depicted at fig. 7. Although the number of ports that can be tunneled via Ethernet is limited by a number of implemented internal ports in this approach, it requires relatively minor changes in the current FPGA-based crossconnect design. Except of the integration of convertor modules, only some changes in the control software will be required. We also expect that the controlling RS-232 console will be converted to Ethernet so that we will be able to control multiple crossconnects from a single control entity, which is useful for centralized creation of distributed virtual WAN topologies.

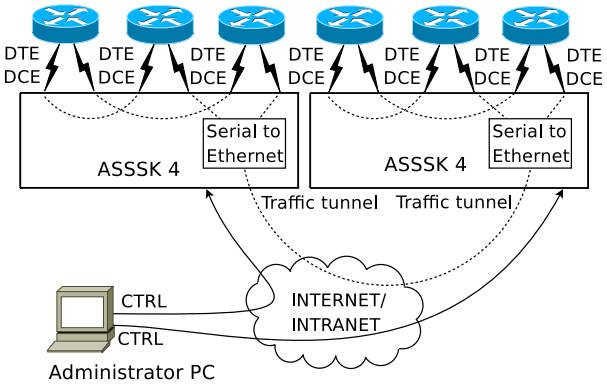

Fig. 7 – Basic Idea of Extension of ASSSK-2 for Frame Tunelling

3. THIRD GENERATION CROSSCONNECTS

As a natural extension of the above mentioned architectures, we proposed a fully-distributed crossconnect architecture, which may prove more flexible and cheaper in the practise. The architecture is based on a big number of small remotely configurable synchronous serial to Ethernet convertors (fig. 8). These convertors are connected to individual WAN ports of network devices (one convertor may potentially handle more than one WAN port). Every convertor may be programmed remotely to which address it has to tunnel PPP/HDLC frames coming from the serial port. Ethernet ports from the convertors of network devices of the lab site will be connected together via standard Ethernet switch. Multiple independent lab sites may be connected together via Internet. The architecture requires a central controller which will create and upload configurations into individual convertors based on the required virtual topology. It is expected that there will be some troubles with passing traffic through firewalls of individual sites, so we plan to include proxy capability into convertor modules so that only a limited number of conduits will have to be configured at firewalls.

The clock for the serial side of the convertor may be provided from router's WAN port or the convertor may generate the clocking for the router

As depicted on fig. 8, our effort is to build a serial-to-Ethernet modules in such a way so that they can also just pass the serial interface signals through. It will allow to have these convertors connected to WAN ports of network devices permanently, so that students will be able to connect network devices physically if necessary during laboratory work. At the same time, we will be also able to connect topology automatically if necessary.

The described approach will also allow to tunnel traffic between Ethernet ports of network devices the similar way as for serial lines, so that we will be able to automatically connect the complete topology with the single technology. The only difference will be the interface type at the router's side of the convertor modules.

The experimental prototype of serial-to-Ethernet modules will be constructed using comercially-available Charon module.

We also feel that the positive side-effect of the fully-distributed architecture is that it will eliminate potential problems with galvanic interconnection of multiple devices connected to a single crossconnect by their serial WAN ports, which may cause problems in some situations.

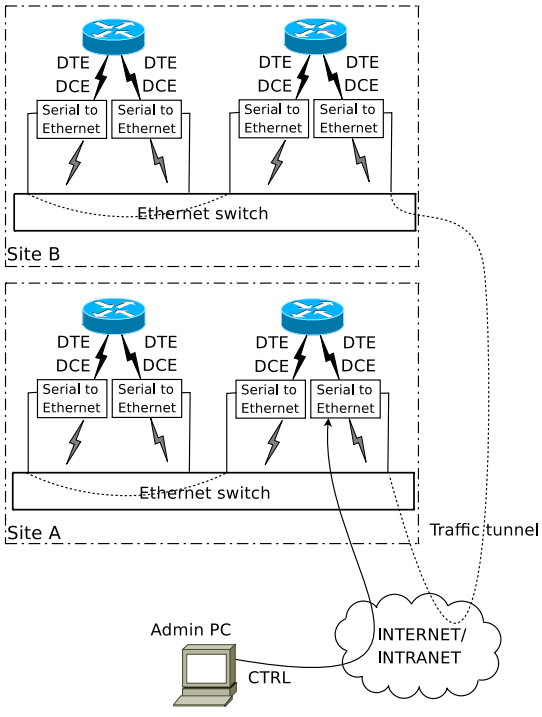

Fig. 8 – The Fully Distributed Crossconnect

4. CONCLUSION

In the article, we presented a couple of architectures and hardware device prototypes for automation of WAN topology interconnection. The approaches presented here may be utilized to make practical education in the networking laboratories more efficient and to support solution of remote laboratory access for the purpose of distant learning. Some of the technologies presented here also form a technological basis of the Distributed Virtual Laboratory [7] developed at our university in cooperation with Silesian University of Opava and with support of Czech Educational Scientific Network (CESNET).

8. REFERENCES

[1] Seidl, D.: System for Automatic Network Configuration Management. Master's Thesis, Faculty of Metallurgy and Materials Engineering, VŠB-TU Ostrava, 2005. [In Czech]

[2] Grygárek, P., Practical Experience with Implementiation of Virtual Computer Network Laboratory and Proposed Ways of its Further Development. Proceedings of Technologies for E-Learning conference, FEL ČVUT Praha, 2006, ISBN 80-01-03512-3, pp.58-68. [In Czech]

[3] Sedlář, P.: FPGA-Based Crossconnect for Serial Lines. Master's Thesis, Faculty of Electrical Engineering and Computer Science, VŠB-TU Ostrava, 2007. [In Czech]

[4] EZL-80 and EZL-80c: converter modules description http://www.hw-group.com/products/sollae/ezl80_en.html [online, April 2007]

[5] Charon 2 Ethernut embedded Ethernet module. http://www.hw-group.com/products/charon2/index_en.html [online, April 2007]

[6] Datasheet of analog switch array http://www.ortodoxism.ro/datasheets/zarlinksemiconductor/zarlink_MT8816_MAR_97.pdf' [online, April 2007]

[7] Grygárek, P., Milata M., Vavříček J., The Fully Distributed Architecture of Virtual Network Laboratory. 5th Int. Prepared for publishing at proceedings of Conference on Emerging e-learning Technologies and Applications, The High Tatras, Slovakia, September 6-8, 2007

THE AUTHOR(S)

Petr Grygárek (Ph.D, MSc.) is a professor-assistant at Department of Computer Science at VŠB-TU Ostrava. His professional interest is focused on computer networking, distributed systems and computer hardware. He is a coordinator and instructor of Regional Cisco Networking Academy and coordinator of distributed virtual laboratory development group. He holds CCNP, CCNA, CCAI and Network Security courses teacher's certificate.

David Seidl (MSc.) is a professor-assistant at Department of Computer Science at VŠB-TU Ostrava. His professional interest is focused on hardware development, low level programming and operating systems. He is a Ph.D. student at Department of Metalurgy and Material Engeneering.