Architectures of Interconnection of Network Laboratories in Distributed Learning Environment

Petr Grygárek [1], Jan Vavříček [1], Tomáš Kučera [1]
[1] Department of Computer Science, VŠB-Technical University of Ostrava, Ostrava, Czech Republic
E-mail: *petr.grygarek@vsb.cz*

Abstract – *For practical laboratory exercises of networking subjects it proved efficient to interconnect laboratories of multiple institutions to build large-scale virtual education environment, which may be accessible as part of common e-learning system. In the article we describe various approaches of creation of such distributed virtual topologies and experience with technical solutions tested in real e-learning education environment. Most of presented technologies are also benefical to automatize topology interconnection process in standard networking laboratory to save teacher's work and gain more time for real students' practising.*

Keywords – *Computer networks, Computer science education, Learning systems*

1. INTRODUCTION

In courses of computer networking it is necessary that student work with real networking devices to gain practical experience with configuring and administration of computer networks. Unfortunately, the cost of networking devices is rather high and the technology advances quickly, which requires a lot of periodical investments into lab equipment to keep it up to date. This is why it is desirable to integrate networking laboratories of multiple teaching institutions and share lab devices. For that purpose, we developed a distributed e-learning system called DVirtlab [1, 2], which allows to automatically create virtual topologies using lab devices of multiple institutions connected via Internet. Students work in the resulting common semi-virtual learning environment remotely, as depicted on fig. 1.

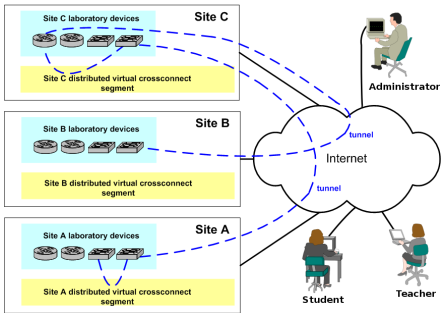

Fig. 1. Transparent integration of networking labs

In scope of the DVirtlab project, we had to develop a mechanism of automatic creation of distributed virtual topologies over the Internet. Our long-term goal was to find a cheap and scalable solution to virtually interconnect pairs of ports of various types, either locally or using tunnelling via Internet. Our results may be helpful for anyone wanting to automatize creation of virtual topologies in either distributed or local learning environment and are presented further in the article.

2. DISTRIBUTED VIRTUAL CROSSCONNECT

In practical implementations of distributed virtual topologies, it is necessary to be able to interconnect pairs of Ethernet and serial (i.e. WAN) ports. To maintain OSI RM layer 3 protocol transparency, we decided to operate whole interconnection system on OSI RM layer 2.

Our goal was to develop an integrated solution capable to interconnect various interface types, even in distributed environment. We named our solution Distributed Virtual Crossconnect (DVC). The world "virtual" indicates that although the solution uses various physical switching elements and interconnection techniques, it may be treated as single entity by rest of the implementation of the e-learning system. Parts of the DVC (called DVC segments) are distributed in individual interconnected laboratories. In essence, implementation of DVC is a mixture of software components (tunnel servers, config generator) and hardware switching devices. The description of required topology is completely independent of types of utilized switching elements.

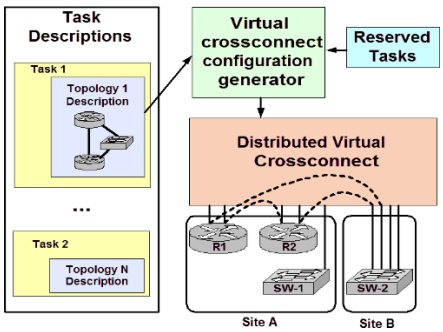

Fig. 2. Integration of DVC into DVirtlab

Fig. 2 shows how is DVC integrated into architecture of our DVirtlab. According to independent description of the distributed virtual topology required for particular task, configuration for all involved switching elements of DVC are automatically

generated and uploaded to switching elements and tunnel servers of DVC segments of affected sites.

During last 3 years, we gained experience with various approaches of DVC implementation. They will be discussed in more details in following sections.

2.1. Interconnection of Ethernet ports

For interconnection of Ethernet ports it proved most efficient and highly extensible to use VLAN-based interconnection approach and exploit standard LAN switches. Using VLAN tunnelling (also called QinQ sometimes because of multiple tagging by IEEE 802.1q headers), we are even able to interconnect trunk links and make switching element to be completely invisible for lab devices and their layer 2 service protocols. VLAN-based approach may be also easily extended to the distributed environment, as shown on fig. 3.

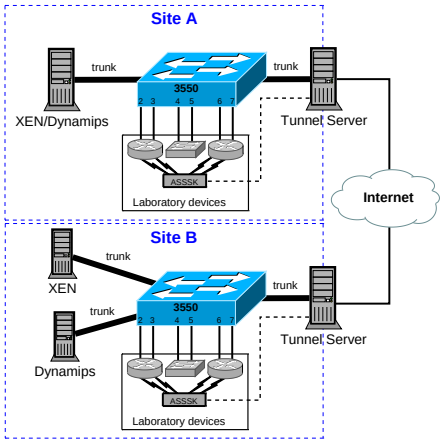

Fig. 3. VLAN-based interconnection approach

To interconnect ports in distributed environment, we implemented a Linux-based Tunnel Server software which receives frames from local lab devices passed through switching element via trunk link and tunnels them in UDP datagrams to the tunnel server of the site where destination lab device resides. The usage of VLANs is also advantageous for incorporation of simulated devices hosted on servers like XEN or DynaMips [3, 4]. Each interface of virtual devices has it's own dedicated VLAN on hosting server connected with the switching element using trunk link. The result is that the rest of DVC does not have to treat simulated devices any other than real ones. Note that there may be arbitrary number of switching elements chained together using trunk links if more ports to connect laboratory devices are needed.

In our very first architecture, we dynamically assigned ports to be connected together to the same VLAN. Although it seemed natural at a first glance, we encountered a lot of technical problems soon. The most important one was that since not all LAN switches implement MIB object to assign ports into VLANs using standard SNMP, we had to by able to handle proprietary CLIs different for each switch model.

Since we needed to interconnect multiple independent virtual topologies in overlapping timeslots in parallel, we also had to explicitly maintain a list of currently used VLAN numbers to be able to allocate non-conflicting ones when a request to create another virtual topology arise. At the end of timeslot we had to remove VLAN assignment from previously connected ports so that no lab device ports stay interconnected unintentionally. Since it was not possible to integrate interconnection configuration intelligence into commercially-available switches, we had to use external control computer to handle configuration of all types of utilized switching elements and watch for periods after which assignment of pairs of ports to the same VLAN should be deconfigured on particular switching element.

Another issue which we had to combat was the dynamic selection of VLANs numbers used in frames tunnelled between tunnel servers of multiple sites in case of distributed virtual topology. Since individual sites of DVirtlab were responsible for generation and uploading of configurations of all DVC switching elements for distributed virtual topologies required by local users, each configuration generator had to carefully select VLANs numbers carried in frames tunnelled over Internet to avoid VLAN conflict.

Because of above mentioned difficulties with handling of VLAN numbers we developed a modified architecture which uses fixed VLANs instead. In that newer architecture, every port of switching element is assigned to unique static VLAN. The switching element is connected to the tunnel server, which handles both internal and external traffic and bridges it across VLANs. All switching element's ports are configured as QinQ tunnel endpoints so that we do not have to differentiate whether we want to interconnect trunk ports or ordinary Ethernet ports of lab devices – the latter is treated as a special case of the former. Although the trunk line between switching element and tunnel server may seem as a bottleneck, the advantages of this solution leverage that drawback. Moreover, in lab tasks which students solve on distributed virtual topologies, only a couple of testing pings and some service traffic obviously passes through topology links. The more heavier load which would use the full capacity of trunk link to tunnel server is very rare in practice.

By integrating the switching intelligence into one component (i.e. the tunnel server) under our own development, we gained a lot of benefits. Since the configuration of switching elements is fixed, we can use all kinds of switch types with various configuration interfaces and freely mix them to create arbitrary cost-efficient switching infrastructure. There is no longer a need of dynamic reconfiguration of individual switching elements and a necessity to handle various configuration methods. The whole switching logic is integrated into the tunnel server, which simplifies the virtual topology configuration considerably. Moreover, we can easily monitor traffic passing through all

virtual topology links on the tunnel server, which allows us to incorporate virtual network probes into the topology [5]. The configuration of the modified DVC is also easier – only configurations for tunnel servers of sites that take a part in the distributed virtual topology have to be generated and uploaded; configuration of switching elements remains fixed.

Although the second version of DVC architecture performed very well in practise, it still suffered with the same drawback as the previous one – the detailed global knowledge of VLANs used in every site is required for generation of DVC configuration. This shortcoming was finally overcame in the most advanced DVC architecture which will be described in section 2.3.

2.2. Interconnection of serial ports

The situation with interconnection of serial ports was more challenging, since there doesn't exist cheap commercially-available device for interconnection of WAN ports, analogous to VLAN-capable LAN switch. This is why we decided to develop a series of our own hardware device prototypes for automatic interconnection of serial ports [6]. Although they differ in the nature of utilized switching core (analog switching array or FPGA-based digital switching core), all of them are designed as centralized solutions for switching of arbitrary number of pairs of connected serial ports. All models are controlled using simple CLI over RS-232 console port.

Although all serial port crossconnects performed very well to automatize creation of topologies in single networking lab, their scalability was limited and they lacked support to create virtual WAN links over Internet. This is why we started to work on fully distributed serial crossconnect architecture (fig. 4).

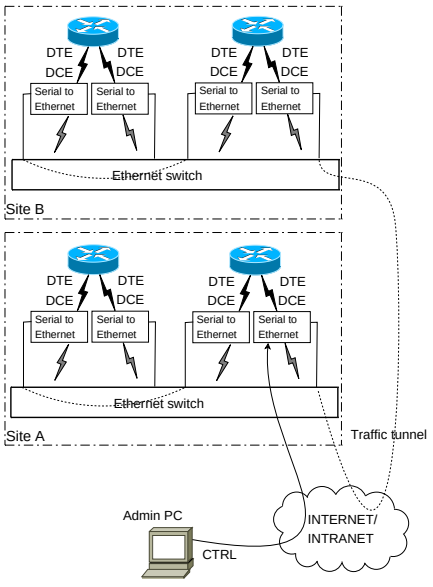

Fig. 4. SE convertors-based crossconnect

This architecture is based on a big number of small remotely configurable bidirectional synchronous serial to Ethernet convertors ("S-E convertors") connected together via Internet and capable of tunnelling of HDLC/PPP frames in UDP. The in-depth explanation of the resulting distributed architecture, which may be also potentially utilized to interconnect Ethernet ports to unify interconnection approaches, can be found in [6].

2.3. Modular DVC architecture

After we started to work on S-E convertors implementation, we had to further modify the DVC architecture to intercorporate them into the current solution and allow to create virtual WAN links across pairs of lab sites. In context of that change, we decided to redesign the philosophy of DVC architecture considerably to compensate for its present above mentioned drawbacks. The resulting architecture is depicted on fig. 5.

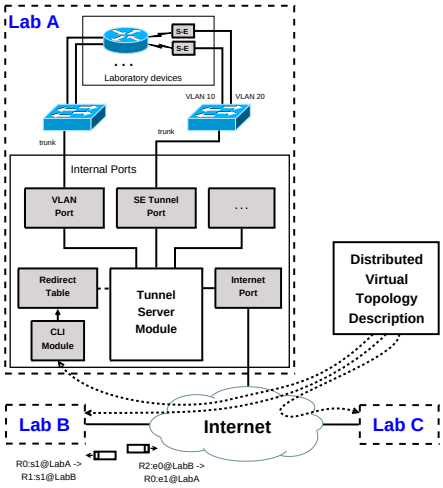

Fig. 5. Modular DVC architecture

The main change against previous versions is a complete encapsulation of techniques used to transport frames between tunnel server and interfaces of local lab devices. The utilized technology is completely contained in respective lab site and hidden from the outside world. Moreover, the new architecture unifies tunnelling approaches for traffic between Ethernet, serial or any other kind of network interfaces. The architecture is highly modular so that new ways of transport of frames between tunnel server and lab devices' interfaces may be added rather easily just by writing a new internal port module. Currently we implemented VLAN Port module which utilizes well-proven VLAN-tunnelling technique and SE Tunnel Port which concentrate traffic from S-E convertors mentioned at section 2.2 (see fig. 5). Tunnel Server module receives frames both from Internet Port and internal port modules and forwards them according to Redirect Table either back to the same module (in case of local interconnection) or to other lab site via Internet Port module. Every internal module is responsible to accompany frame passed to Tunnel Server module with designation of source interface in

the unified text-based format (device:interface@site). Tunnel server module then translates source interface name to the destination interface name using local Redirect Table and if the source site corresponds to destination one, it sends the frame back to the Port module from which it originally came. Otherwise the Tunnel Server module sends frame to the Internet Port to tunnel it to the remote site. In both cases, Tunnel Server module accompanies the frame with destination interface name corresponding to the source interface name according to Redirect Table. The destination internal port module uses the destination interface name to search appropriate destination VLAN or S-E convertor's address, whereas Internet Port just includes it into the envelope of tunnelled frame. Similarilly, the mapping between destination interface names and communication parameters necessary to send frame to appropriate lab device interface using technique handled by particular Port module is contained completely in the configuration of the respective port module. The configured mapping is also used to determine the global name of the lab device interface from which the frame processed by internal port module originated.

Records of Redirect Table of individual tunnel servers are added or deleted using Command Line Interface accessible remotely via TCP connection. The entity that controls creation of distributed virtual topologies just sends a list of pairs of names of interfaces to be connected to CLI modules of tunnel servers of labs involved in the topology. To configure interconnection using CLI manually, Telnet client may be also used.

The key feature of the architecture is that frames tunnelled over Internet have the same encapsulation format for all types of local interconnection techniques. Unified text-based envelope is added to every frame tunnelled in UDP packet so that global names of both source and destination interfaces completely define source and destination of every frame. The advantage of a such approach is that lab sites may use arbitrary and mutually different interconnection techniques, provided that they adhere the common encapsulation format between tunnel servers. The parameters of local mechanism to transport frames on local side (e.g. VLAN numbers) are completely contained in respective tunnel server module and no information concerning local interconnection system has to be published globally anymore. It is also much more easy to trace frames passed either between lab devices of the same lab site or between sites, which simplifies debugging of DVC setup considerably. Moreover, virtual network probes mentioned above can be now installed also into virtual serial lines. We even consider a quite easy implementation of simulation of link outages and flapping to let students gain enough experience with real network troubleshooting.

3. CONCLUSION

In the article we presented a couple of technologies for creation of distributed virtual nework topologies which we developed so far and our experience gained during periods of their operation. Ability to interconnect topologies automatically is very benefical for creation of e-learning environments focused on computer networking because it considerably reduces cost of operation of large-scale distributed environments for practical education. The advantages of remote teaching in that environment were published in [2].

Many interconnection approaches presented above may be also useful in local networking laboratories. The automation of topology interconnection process may save a lot of time and helps to avoid mistakes and device's mechanical depletion comparing with manual interconnection of network topologies before each exercise.

REFERENCES

[1] P. Grygárek, M. Milata, J. Vavříček, "The Fully Distributed Architecture of Virtual Network Laboratory", *Proceedings of 5th International Conference on Emerging e-Learning Technologies and Applications*, Vol. 1, Stara Lesna, Slovakia, Elfa s.r.o., September 2007

[2] P. Grygárek, M. Milata, "Piloting Environment of Distributed Virtual Networking Laboratory", *Conference Virtual University*, Vol. 1, Slovak Technical University, Bratislava, Slovakia, December 2007, pp. 209-212

[3] The Xen^{TM}virtual machine monitor, Available: http://www.cl.cam.ac.uk/research/srg/netos/xen/

[4] DynaMIPS project – Cisco 7200 Simulator, Available: http://www.ipflow.utc.fr/

[5] R. Novák, "Implementation of Traffic Monitoring on VLANs at Virtual Network Laboratory using tethereal or tcpdump", Bachelor's Thesis, Faculty of Electrical Engineering and Computer Science, VŠB-TU Ostrava, May 2008

[6] P. Grygárek, D. Seidl, "Automatic WAN Topology Interconnection and it's Usage in CNAP Networking Laboratories", *Proceedings of 5th International Conference on Emerging e-Learning Technologies and Applications*, Vol. 1, Stara Lesna, Slovakia, Elfa s.r.o, September 2007