DiProNN: VM-based Distributed Programmable Network Node Architecture

TERENA Networking Conference 2007, 21.–24. May 2007, Copenhagen, Denmark

Tomáš Rebok
xrebok@fi.muni.cz

Faculty of Informatics
Masaryk University Brno
Botanická 68a, Brno 602 00
Czech Republic

CESNET z. s. p. o.
Zikova 4
160 00 Praha 6
Czech Republic

Introduction

The active network approach allows an individual user to inject customized programs into active nodes in the network, usually called programmable/active nodes, and thus process data in the network as it passes through. As the speeds of network links still increase, and subsequently, the applications' demands for the network bandwidth increase as well, a single active node is infeasible to process such high-bandwidth user data in real-time, since the processing may be fairly complex (e.g., high-quality video down-sampling for videoconference clients with low bandwidth connectivity).

The main goal of our work is to present the architecture of DiProNN node—the VM-based Distributed Programmable Network Node, that can improve the scalability of such an active system with respect to number of active programs simultaneously running on the node and with respect to the bandwidth of each passing stream processed. Furthermore, to make DiProNN programming more comfortable, we also propose suitable modular programming model that takes advantages of DiProNN virtualization—using standard network services the DiProNN node interconnects standalone special-purpose active programs into an abstractly described complex processing system.

DiProNN: Distributed Programmable Network Node

DiProNN architecture assumes the infrastructure as shown in Figure 1. The computing nodes form a computer cluster interconnected with each node having two connections:

- one *low-latency control connection* used for internal communication,
- at least one *data connection* used for receiving and sending data.

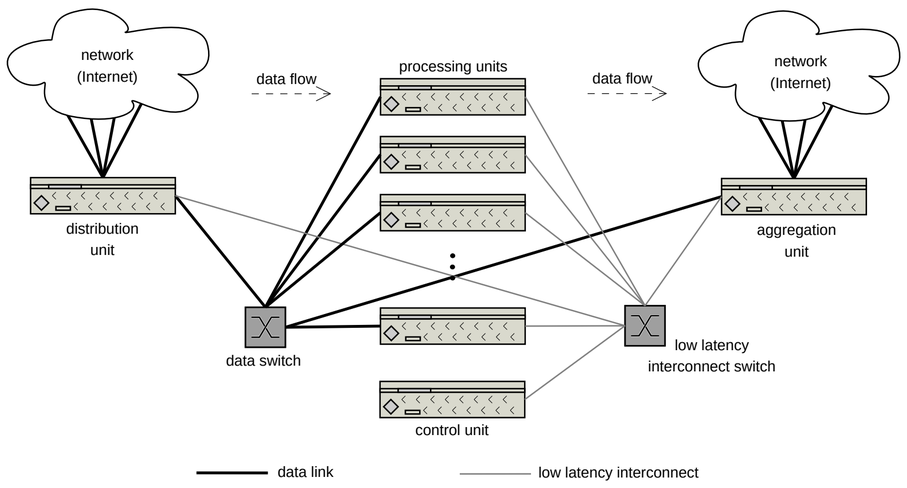

Figure 1: The DiProNN architecture.

As a communication protocol, the DiProNN uses the transmission protocol called *Active Router Transmission Protocol (ARTP)* we designed. The ARTP is a connection oriented transport protocol providing reliable duplex communication channel without ensuring that the data will be received in the same order as they were sent.

DiProNN Processing Unit Architecture

DiProNN Processing Unit is designed to facilite its implementation based on virtual machines (VM), that enable:

- DiProNN users to upload not only standalone active programs, but also the whole virtual machine with its operating system and with set of active programs running inside,
- DiPRoNN administrators to run their own set of fixed virtual machines, each one with different operating system and generally with completely different functionality (providing fixed functionality and/or serving as an execution environment, where active programs without their own VMs are executed),
- strict scheduling of DiProNN resources to individual VMs, e.g., CPU, memory, and storage subsystem access.

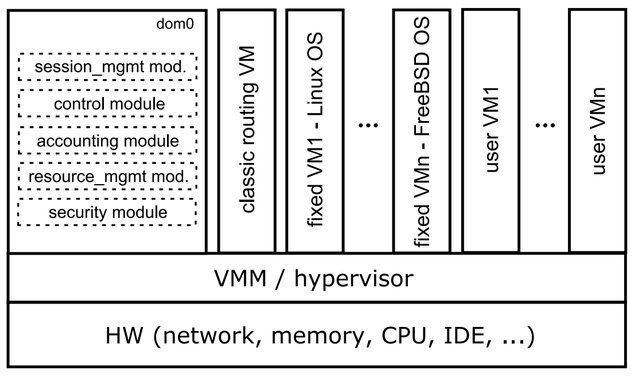

Figure 2: The DiProNN Processing Unit architecture.

DiProNN Programming Model

The DiProNN programming model is based on workflow principles—it enables the complex functionality to be constituted using simple processing blocks. In our case, the processing block is an active program and the communication among such active programs is thanks to the virtualization mechanisms provided by machine hypervisor using common network services. To achieve desired level of abstraction all the active programs as well as the input/output data/communication interfaces are reffered by their hierarchical names as shown in following example.

Example

Consider following situation: there is one incoming high-bandwidth video stream (e.g., an HD stream having 1.5 Gbps) and one high-quality audio stream (both transferred using ARTP protocol). In the DiPRoNN, both streams must be transcoded into low quality streams for specified set of clients behind low-bandwidth lines, and for some clients the streams must remain in the original quality. At the output, there must be both audio and video streams of given quality mixed into just one output stream (thus having two output streams—one high quality and one low quality) and the precise time synchronization between audio and video in both output streams is also required.

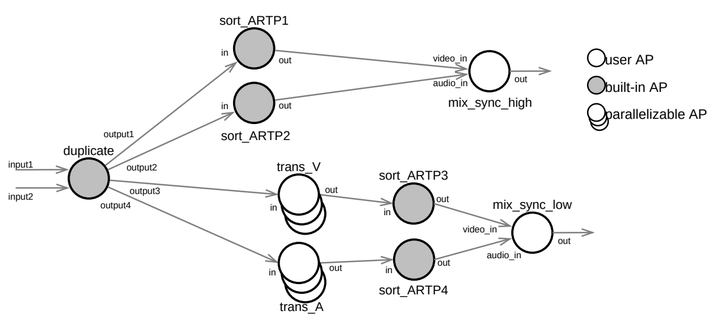

Figure 3: The example of DiProNN Session Graph and its DiProNN program.

```
Project first_project.HD_transcode;
# project parameters (owner, notifications,
# overall resource requirements, ...)
{AP name="sort_ARTP1" ref=localService.sortARTP;
  # AP parameters
  inputs = in;
  out = my_VM.mix_sync_high.video_in;
}
{VM  name="my_VM" ref=my_VM_image.img;
  # VM parameters
  {AP name="mix_sync_high" ref=mixer_syncer;
    inputs = video_in, audio_in;
    precision = 0.001;  # 1ms
    output = DiPRoNN_OUT;
        # special output---DiPRoNN node output
  }
  # other APs ...
}
# other VMs/APs ...
```

Acknowledgements and References

Acknowledgements

This project has been supported by a research intent "Optical Network of National Research and Its New Applications" (MŠM 6383917201), "Parallel and Distributed Systems" (MŠM 0021622419), and "Integrated Approach to Education of PhD Students in the Area of Parallel and Distributed Systems" (No. 102/05/H050).

References

- T. Rebok: *VM-based Distributed Active Router Design*, MEMICS 2006, Mikulov, Czech Republic, 2006.
- T. Rebok: *Active Router Communication Layer—protocol ARTP*, CESNET z. s. p. o., technical report, 2004.