PEER TO PEER SYSTEM DEPLOYMENT

Anton BALÁŽ, Norbert ÁDÁM
Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice,
Letná 9, 042 00 Košice, Slovak Republic, E-mail: {anton.balaz, norbert.adam}@tuke.sk

ABSTRACT

The aim of the following article is to create system for automatic deployment of an operating system, which enables to deploy operating system from the network with priority of throughput and data transfer efficiency. There are lots of software solutions but majority of them use multicast for data transfer efficiency which is, very often, disabled in many network topologies. This led us to the intention of design which will applies peer-to-peer communication. Proposed system architecture utilizes peer-to-peer communication between nodes that leads to increasing throughput while deploying multiple systems OS, even through cloud services.

Keywords: *system deployment, network, operating system, peer-to-peer*

1. INTRODUCTION

Computers could be hardly used without any operating system. Silberschatz, Galvin and Gagne [1] define operating system(OS) as a program, which controls computer hardware. OS represents an interface between user and computer hardware. In this context, we can view an operating system as a resource allocator. The aim is to create solution which enables the user to execute programs efficiently.

OS installation by installation CD or USB drive is usually not time consuming. However, if we add drivers installation and installation of an additional software, the time could reach up to several hours. For organizations, installing hundreds of computers per day will increase total cost of ownership significantly. A solution for this problem is software for massive OS deployment which installs several computers at the same time.

The aim of this article is to design system for massive OS deployment which will utilize PXE protocol and peer-to-peer communication. During peer-to-peer communication, nodes are equivalent and with increasing number of nodes data throughput increases as well, since a node can transfer data from several nodes at the same time. In contrast with standard client-server topology, with an additional connected node, the data throughput decreases, since nodes share server's constant data bandwidth.

2. STATE OF ART

The most appropriate pattern for massive OS deployment is PXE protocol adaptation. The main task of this protocol is client preparation, for instance including OS installation. PXE protocol works as follows:

1. Client sends DHCP request.
2. DHCP server replies with list of PXE servers.
3. Client selects boot server and sends request.
4. Boot server answers with boot file name.
5. Client downloads executable file.
6. Client is able to verify downloaded file or execute the file.

List of deployment software includes FOG Project, Clonezilla Server Edition, Acronis Snap Deployment or Symatec Ghost Solution. All mentioned software utilizes multicast or unicast communication. Multicast enables to send packets to group of clients in a network using special type of network address – multicast group address. Network equipment forwards received packets to all members of the multicast group.

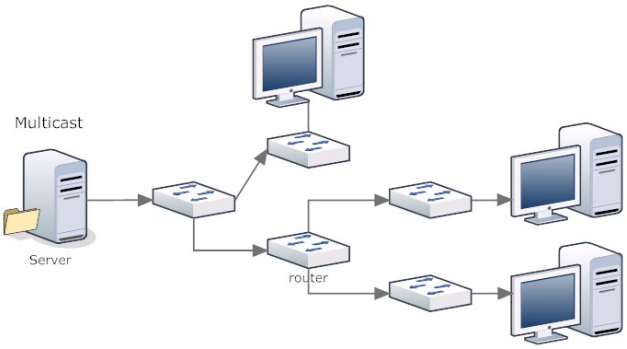

Fig. 1 Multicast Communication

As seen in figure 1, multicast enables to decrease network utilization while sending only one data stream which is replicated by network equipment. Williamson [3] defines several problems regarding multicast. One could be high network equipment load. Figure 1 depicts how the first router creates two data streams. If a router is not able to efficiently replicate data streams, problems may occur with higher number of receivers.

Next, multicast drawbacks are unreliable packets delivery. Multicast applies UDP protocol. Multicast applications must deal with occasional data lost.

Problem could occur in redundant network topology where there are several routes to a receiver. While router forwards data to multiple network interfaces, destination could receive the same data multiple times. If we have application controlled by multicast messages, the same message could be performed several times.

Peer-to-peer communication gets attentions thanks to Napster system which shares music files. Intel employees define peer-to-peer as resources sharing service between

systems – Fig. 2. Based on this, two characteristics of peer-to-peer could be deduced [4]:

- Scaling – Peer-to peer does not have an algorithm or technical constraints of the system size, complexity is constant no matter the number of the system nodes.
- Reliability – System functionality is not dependent from all the nodes, failure of any node will not lead to the entire system malfunction.

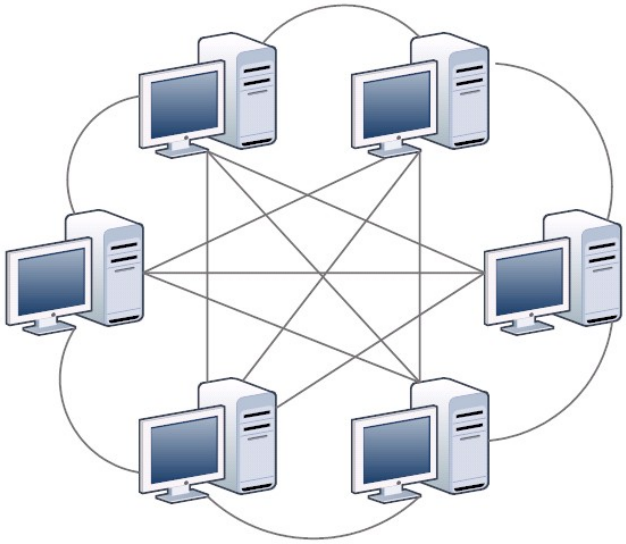

Fig. 2 Peer to Peer Network Topology

Peer-to-peer network provides several protocols where everyone has its utilization. The most widespread one is BitTorrent. This protocol is used for delivering large-size files between end-users. During data transfer, the files are divided to smaller parts. Automatically after some part is already downloaded, this part is forwarded to other clients who speed up the process of data delivery.

In Kunji work [5] BitTorrent protocol is described as peer-to-peer network with centralized topology with some differences. BitTorrent protocol does not provide service for searching. Data are localized through index served by central server.

In BitTorrent network, two types of clients occur:

- Client (downloading, sharing)
- Seeder (sharing)

Clients are end users who do not have any or some parts of the file. Seeders are clients who have already downloaded the entire file and are active for seeding data to other clients. After downloading the entire file, clients are defined as seeders. To download a file, there has to be at least one seeder [6].

Next important component is tracker. Its task is to maintain information about active clients. Clients are sending information about files being downloaded. Tracker replies with list of active clients who share the file. Tracker does not share data [7].

Downloading is not possible without the file which includes information about file name, file size, hash data and tracker address. This file is compulsory for every client.

Downloading steps [8]

1. Client downloads torrent file from web server or other source.
2. Client contacts tracker with request for list of active clients for that torrent.
3. Tracker replies with list of clients.
4. Client sends request for file parts to active clients/seeders.
5. If client accepts request, downloading starts.

While sharing by BitTorrent protocol, the file is divided to smaller parts of the same size. After client downloads some part, SHA1 hash is computed and compared to the value with the one in the torrent file. If the values match, client replies to other clients with availability of that file part.

BitTorrent protocol utilizes TCP connection. To maximize capacity of TCP connection, streaming technique is applied. Smaller single parts are divided to even smaller ones, usually 16kB. These subrequests are queued and next incoming data block executes a request for the next data block. This ensures continual TCP connection utilization.

BitTorrent capacity could be described by deterministic model [5] which focuses on the situations when high number of clients with minimal number of seeders occurs. This model assumes that $n = 2^k$ clients are trying to download parts of s bits. It has to be transferred to *ns bits*. The best strategy is to transfer this part to another user with speed of b whereby the transfer capacity increases twice. By this, client downloads one part every $\tau = s/b$ seconds, thus every τ seconds the capacity increases twice, leading to an exponential growth of $2^{\frac{t}{\tau}}$ in the unit of number of peers available to serve others. By this, selected part of the file is delivered to clients after $log_2 n = k$ seconds. Average downloading latency of client is $\bar{d}$.

$$\bar{d} = \frac{1}{n}\sum_{j=1}^{n} d_j = \sum_{i=1}^{k-1} 2^{i-k}\tau(i+1) = k\tau - \frac{n-1}{n}\tau = \\ = \tau(log_2 n - \frac{n-1}{n}) \approx \tau log_2 n \quad (1)$$

3. DESIGNED SYSTEM DEPLOYMENT

The proposed system architecture consists from two parts. One represents PXE server, main management part. Second one is BitTorrent client, all required components for file sharing.

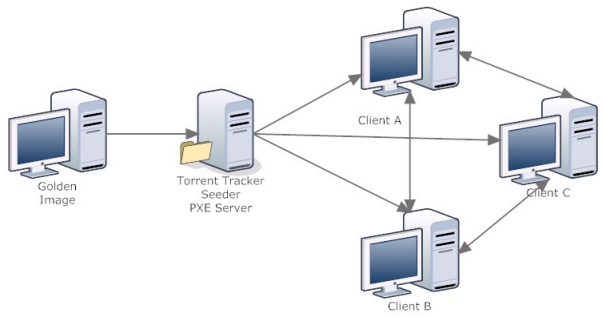

Fig. 3 System Architecture

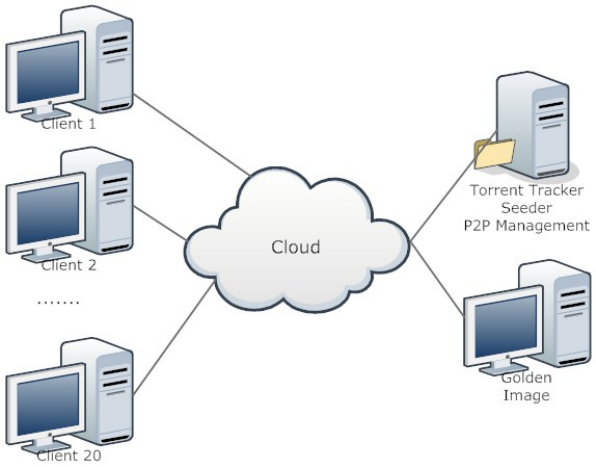

Fig. 4 Evaluation Topology

BitTorrent part of the system consists of the tracker, torrent client and file of an OS image, all running on Linux OS. Linux was selected for required access to all parts of the golden image file system. Main function of this component is to share cloned OS data. Actual version of the system deploys only present MS Windows OS family. Implicitly, this component is serving as seeder, tracker and data storage of golden images.

System management is created from the modified Linux distribution System Rescue CD with added torrent client. This component is serving as PXE Server, DHCP and TFTP server. Task of PXE server is to provide boot kernel to deployed system, DHCP for IP address assigning, TFTP for initial kernel booting. Proposed architecture is based on modified System Rescue CD. This customized distribution includes utilities and scripts for the golden image preparation and its sharing.

System deployment is performed by a set of scripts which:

- Create file systems,
- Download torrent files, and
- Restart newly cloned OS.

4. PERFORMANCE EVALUATION

Performance evaluation is based on the comparison of a time needed for OS deployment between designed system and Clonezilla Server Edition. The testing scheme consists of testing pattern of OS – golden image. Groups of end-systems intended for group clonning of OS and system for managing of P2P data transfer. The scheme is formed out of heterogenous network of intranet and Internet. Therefore, while testing, data transfer of Unicast type was compared to the proposed P2P – Fig. 4.

First step was to create golden image of an OS - MS Windows 7. After preparation, evaluation was performed on the tested hardware. During system PXE booting, deployed hardware downloads installation script and executes the following steps:

1. NTFS file system creation.
2. Download torrent file through TFTP protocol.
3. Download golden image (P2P).
4. Deploying image to storage.
5. System restart.

Deployment speed was measured from the boot to restart of the system. Obtained data are recorded in table 1.

After initial evaluation, the test with multiple deployed systems was performed. Data are recorded in table 1.

Next, the alternative software was evaluated – Clonezilla SE. Again, the time needed for OS deployment with one and multiple deployed systems was measured and data are the table 1.

Table 1 Deployment Time

Deployment SW	No. HW / Time	No. HW / Time
Designed arch. (P2P)	1 / 134min	20 / 73min
Clonezilla SE (Unicast)	1 / 110min	20 / 533min

From the measured results – designed architecture deploying one system is behind competition, but cloning systems are employed mainly for massive OS deployment. And here designed architecture is achieving better results.

5. CONCLUSION

The aim of this work was to create a system for deployment of operating systems which would not utilize multicast for data transfer but peer-to-peer communication. After review of some peer-to-peer protocols, BitTorrent was proved to be the most appropriate. This is because of its methods of client and parts selection what significantly contributes to more effective data transfer.

In sec. 4, final architecture of the cloned system was subjected to bandwidth tests. Due to restrictions in network infrastructure, it was not possible to utilize multicast during the tests, thus the tests utilized unicast. The tests revealed that system using BitTorrent is much more effective when deploying higher number of operating systems as unicast communication, especially in network bandwidth. Moreover, the test revealed benefits of the created systems in a wider context, since BitTorrent protocol is not as restricted as multicast and it allows transferring data through Internet as well.

ACKNOWLEDGEMENT

This work was supported by the Slovak Research and Development Agency under the contract No. APVV-0008-10 and project KEGA 077TUKE-4/2015 Promoting the interconnection of Computer and Software Engineering using the KPIkit.

REFERENCES

[1] SILBERSCHATZ, A. – GALVIN, P. B. – GAGNE, G.: Operating System Concepts. In: R. R. Donnelley, Jefferson City, 2008, pp. 3-6.

[2] WITTMANN, R.: Multicast Communication: Protocols, Programming and Applications. In: Academic Press, San Diego, 2000, pp. 1-20.

[3] WILLIAMSON, B.: Developing IP Multicast Networks. In: Cisco Press, 2000, [online] [2015-5-17] http://docstore.mik.ua/cisco/pdf/routing/Cisco.Press.CCIE.Developing.Ip.Multicast.Networks.pdf

[4] Yu-Kwong R. KWOK: Peer-to-Peer Computing, Applications, Architecture, Protocols, and Chalenges. In: CRC Press, Boca Raton, 2011, pp. 1-14.

[5] KUNJIE, X.: Performance Modeling of BitTorrent Peer-to-Peer File Sharing Networks. In: The University of Pittsburgh, [online] [2015-5-17] http://arxiv.org/ftp/arxiv/papers/1311/1311.1195.pdf

[6] TOOLE, R.: BitTorrent Architecture and Protocol. In: The University of Massachusetts, Dartmouth, 2006, [online] [2015-5-17] https://robot.bolink.org/ebooks/BitTorrent%20Architecture%20and%20Protocol.pdf

[7] GUO, L. – CHEN, S. – XIAO, Z. – TAN, E. – DING, X. – ZHANG, X.: A Performance Study of BitTorrent-like Peer-to-Peer Systems. In: IEEE Journal on selected areas in communications, 2007, [online] [2015-5-17] https://web.njit.edu/~dingxn/download/BT-JSAC.pdf

[8] TIAN, Y. – WU, D. – NG, K. W.: Modeling, Analysis and Improvement for BitTorrnet-Like File Sharing Networks. In: The Chinese University of Hong Kong, [online] [2015-5-17] http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4146984

Received February 16, 2016, accepted April 24, 2016

BIOGRAPHIES

Anton Baláž (MSc., PhD.) was born in Sobrance, Slovakia, in 1980. He received the master degree in Informatics in 2004 from Faculty of Electrical Engineering and Informatics, Technical University of Košice. In 2008 he received PhD. in field of Computer Security. Since 2007 he is working as Assistant Professor at the Technical University of Košice.

Norbert Ádám (MSc., PhD.) was born in 1980. He graduated (MSc.) with distinction at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice in 2003. He defended his PhD. in the field of Computers and computer systems in 2007; the thesis title was "Contribution to simulation of feed-forward neural networks on parallel computer architectures". Since 2006 he is working as a professor assistant at the Department of Computers and Informatics. He became Head of the Computer Architectures and Security Laboratory at the Department of Computers and Informatics in 2008. His scientific research is focusing on the parallel computers architectures.