COMMON PROTOCOL FOR DISTRIBUTED NETWORK FILE SYSTEM

P. Peniak [1)], F., Kallay[2)]

[1)], *IT service center for Easter Europe (SCEE) Continental AG, Continental Matador Truck Tires s.r.o*
T.Vansovej 1054/45, 020 01 Púchov, tel.: +421 42 4613479,e- mail: peter.peniak@conti.sk

[2)], *Faculty of Electrical Engineering, University of Žilina,*
Univerzitná 8251/1, 010 26 Žilina, e- mail:fedor.kallay@fel.uniza.sk

Summary Paper deals with common protocol for distributed network file system. Focus is on CIFS protocol from Microsoft, the enhanced version of Microsoft Server Message Block (SMB), that is proposed as possible common solution for file sharing among distributed systems. There are new requirements included as well, that are to be implemented due to recent changes in applications and devices and has been addressed in new generation of distributed file system protocols such as AFS2, CODA and WebDAV.

1. INTRODUCTION

A network file system is defined as any computer file system that supports sharing of files, printers over a computer network. Network file services enable computers attached to the network to access files stored on other computers in the same way that they access files on their own disks. That is, the client interface used by programs should not distinguish between local and remote files. It is up to the network file system to locate the files and to arrange for the transport of the data [1], [2].

The computer which provides access to its files and directories is called file server, and the computer using these files and directories is the client. A computer can be client and server at the same time. For communication between servers and clients, the specific network protocols are used. The most popular are the following protocols, see Tab.1:

Tab.1 Network file system protocols

–	Novell NetWare	DoD TCP/IP	Microsoft Network
7 6	NCP	FTP, HTTP, NFS	SMB, RPC
5			NetBIOS
4	SPX	TCP/UDP	NetBEUI
3	IPX	IP	

- NFS(Network File System) –SunSystems, IETF
- NCP (NetWare Core Protocol) -Novell
- Server Message Block (SMB) –IBM, Microsoft
- File Transfer Protocol (FTP) -IETF
- Hypertext Transfer Protocol (HTTP) -IETF

These protocols were implemented in the different operation systems such as Microsoft Windows (SMB) and UNIX/LINUX (NFS) [3]. Therefore, their clients cannot use the different network fileservers by default, as shown in Fig.1a). There are multi-client solutions applied, with several implemented network file system protocols within given operation system. Fig.1b) shows multiclient configuration, with the protocol stacks for NFS and SMB (SAMBA implementation of SMB server in LINUX) so users who work with UNIX or Microsoft operation systems, are able to use the both network file systems. Another option is to develop and implement a common network file system that would be supported by all platforms and suppliers. Then, just one protocol stack could be used for access and sharing files via Internat. This option is shown in Fig.1c.

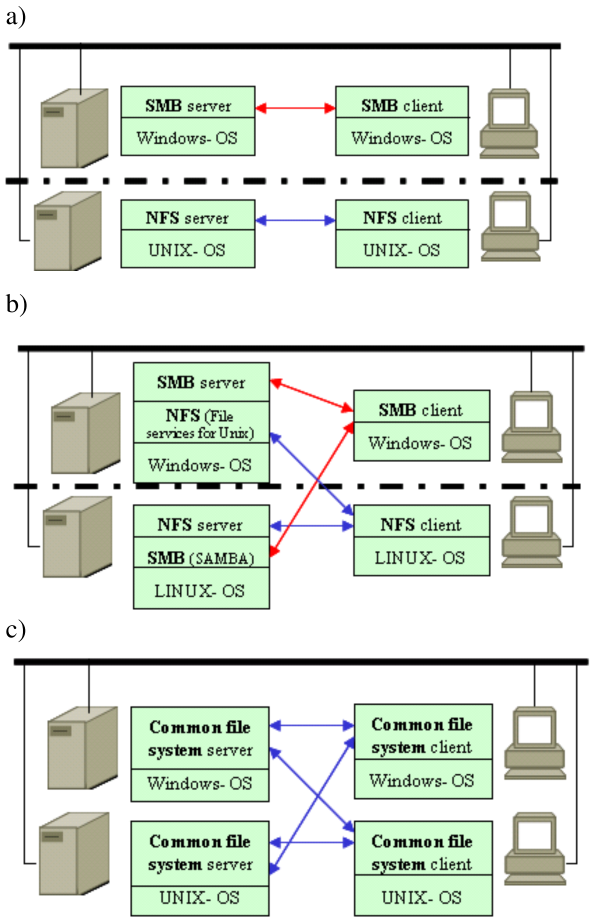

Fig.1 Possible access to network file systems

Due to Microsoft superiority in number of installed nodes in Internet, the introduction of common network file system protocol will require its strong support and acceptance. Microsoft is currently trying to establish its protocols as the official standard of the Internet. It could perspective replace the currently used FTP protocols and file transfers through the HTTP protocol, which represents substantially heterogeneous environment with problematic security. Microsoft has been trying to push Common Internet File System (CIFS), protocol through standardization process since 1996 [4]. CIFS drafts were even provided to IETF so that it could be approved among Internet stadard protocols.

2. COMMON INTERNET FILE SYSTEMS

Common Internet File Systems (CIFS) is the enhanced version of Microsoft Server Message Block (SMB) protocol that defines a remote file-access to share data on local disks and network file servers, among distributed systems across intranets and the Internet. In addition to SMB, which was developed on the top of NETBIOS-API, CIFS is able to run over TCP/IP and utilizes the Domain Naming Service (DNS). It is optimized to support communication via Internet with various speed levels. CIFS complements Hypertext Transfer Protocol (HTTP) while providing more sophisticated file sharing than traditional protocols, such as FTP.

The CIFS protocol is not only available for servers and PCs, but also for embedded devices and distributed systems. It had to be scaled down in size and reworked for multitasking features to meet the needs of embedded devices such as real-time, deterministic response times, small memory, and a variety of microprocessor–file system combinations. The using of CIFS for servers and embedded devices is shown in Fig.2. The file sharing is independent on hosting platform and embedded operation system/file system.

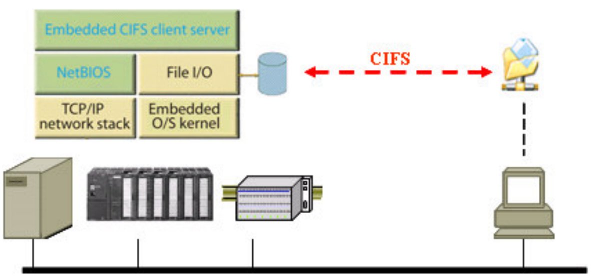

Fig. 2 CIFS protocol in servers and embedded devices

CIFS provides the following functions:

❶ **File Access**

- Support of basic file operations (open, close, read, write and seek).
- Record lock and unlocking for multiple clients access and file sharing and locking.

❷ **Performance and portability**

- Highly integrated within the operating system.
- Support for all recent Microsoft platforms and other operation systems such as Unix, VMS, IBM, Macintosh

❸ **Security**

- Support of anonymous access(no authorization).
- Enhanced security and authenticated access

❹ **Optimization**

- Improved protocol performance
- Reduction of overhead in SMB protocol

❺ **Global File Names**

- No need to mount remote file systems,
- Access based on globally significant names

❻ **Unicode File Names**

Because CIFS is mainly based on SMB protocol we will explain below its core functions.

3. SMB PROTOCOL

The **SMB** (***Server Message Block***) protocol was originally developed by the company IBM in co-operation with the company Microsoft for one of the first network operating systems, the ***PC Network Program v.1.0***. Today, this protocol is the most frequently used one in the field of peer-to-peer communication of LAN network file servers and clients. The SMB protocol is currently used by file and print servers IBM, Microsoft's network operating systems (Lan Manager, Windows NT, XP, Vista). The protocol header and basic services are shown in Tab. 2 and Tab. 3.

Tab. 2 Header of SMB protocol

SI D	S	C	R C	A H	RC	RS	NI D	PI D	UID	MI D	P	PC d	BL	B
1	3	1	1	1	2	15	2	2	2	2	1	2	2	1

Σ 38 B

MARKING		MEANING
SID	*SMB **Identification***	Identification of SMB protocol (0xFF)
S	*Server*	Identification of SMB server's dialect
C	*Command*	Functional code of called SMB service
R	*Return Class*	SMB function return code class
AH	***Processor AH register***	Result of operation in register of AH processor
RC	*Return Code*	Return code of operation
RS	*Reserve*	Reserve for future extension
NID	*Net Path **ID***	Identifier allocated to shared means
PID	*Process **ID***	Identifier of client's process
UID	*User **ID***	Identifier of user
MID	*Multiplex **ID***	Multiplex identifier of client process
Prmct	*Parameter Count*	Number of optional parameters for called function

PC	***Parameter Code***	Called function parameter code
BL	***Buffer Length***	Length of data part of SMB block
B	***Buffer***	First octet of data part of SMB block

Tab.3 Example of SMB services

SMB BLOCK	CODE	SERVICE
SMB_COM_CREATE_DIRECTORY	0x00	Creation of directory
SMB_COM_DELETE_DIRECTORY	0x01	Deletion of directory
SMB_COM_OPEN	0x02	Opening of file
SMB_COM_CREATE	0x03	Creation of directory
SMB_COM_CLOSE	0x04	Closing opened file
SMB_COM_DELETE	0x06	Deletion of file
SMB_COM_RENAME	0x07	Renaming of file
SMB_COM_READ	0x0A	Reading from an opened file
SMB_COM_WRITE	0x0B	Writing into opened file
SMB_COM_LOCK_BYTE_RANGE	0x0C	Locking file byte range
SMB_COM_UNLOCK_BYTE_RANGE	0x0D	Unlocking file byte range
SMB_COM_CHECK_DIRECTORY	0x10	Searching through directory
SMB_COM_SEEK	0x12	Transfer in open file
SMB_COM_TREE_CONNECT	0x70	Connection to shared means
SMB_COM_TREE_DISCONNECT	0x71	Disconnection from shared means
SMB_COM_NEGOTIATE	0x72	Negotiation of SMB parameters
SMB_COM_SESSION_SETUP_ANDX	0x73	Connection to SMB server
SMB_COM_LOGOFF_ANDX	0x74	Disconnection from SMB server
SMB_COM_OPEN_PRINT_FILE	0xC0	Opening print file
SMB_COM_WRITE_PRINT_FILE	0xC1	Writing to print file
SMB_COM_CLOSE_PRINT_FILE	0xC2	Closing print file

The SMB block with a client's request and a server's answer are practically identical, they just differ in the data part and the length of the data buffer. An example of communication of a client with an SMB server can be seen in Fig. 3.

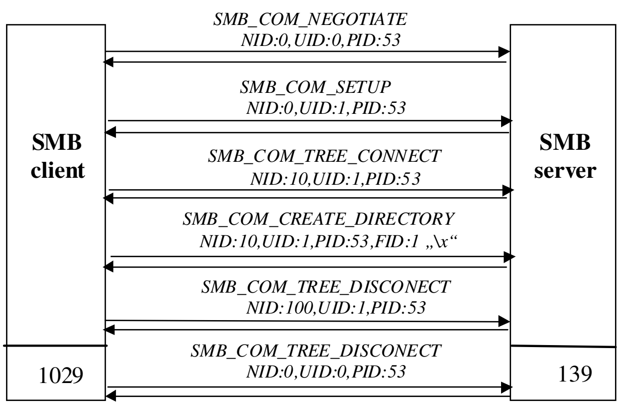

Fig.3 Example of client communication with SMB server

The protocol works on the basis of the client/server model. A server provides to network clients the co-called shared resources, usually shared disks, directories, print files and named pipes. Shared resources are identified through an universal network address called **UNC**

server_name *resource_name.*

A client secures the formulation of requirements of shared server resources. The SMB server analyses requirements sent by a client in the form of an SMB block (packet), it checks an access permission, and based on that it carries out a required operation (file opening, writing into print file, creation of a directory, etc.) The reply, together with the result, is sent to the client in an identical SMB block. In the implementation of SMB servers, two different methods of securing access to shared means of the server are offered through:

- ❶ control of access at the share level
- ❷ control of access at the user level

In the first case the access to shared resources is allowed to all network nodes that have a valid password allocated to a resource of a server. During an attempt to access any server resource, a password is required from the client.

After its successful entering, an NID (Network ID) resource identifier is allocated to the client, through which it accesses the resource. Thus a number of passwords with different access levels can be allocated to each shared server resource. In the other case, the client is verified though a name and a password immediately after the connection to the server, and in the event of their correctness, the server allocates to the client its user identifier **UID** (User **ID**), through which the server derives access rights to individual shared resources. In a multi-user environment, when several processes can communicate simultaneously with the server, a couple of PID (***Process ID***) and **MID** (***Multiple ID***) identifiers can serve for their distinguishing.

The client sends the server an SMB_COM_NEGOTIATE request, through which the server and the client negotiate connection parameters and a supported protocol version. Subsequently, the client sends to the server a SMB_COM_SETUP request, in which there is the user's name and password. If the server works in the user-level mode, it allocates an UID identifier to the client. Through the SMB_TREE_CONNECT message the client connects to a shared directory with a TID identifier allocated by the server. Then the creation of a (\x) FID subdirectory and the disconnection of the server follows.

```
Ethernet:   (006097ef0da6 -> WDgtl 23dedf)  type: IP(0x800)
Internet:       170.1.1.10 -> 170.1.1.3       hl: 5   ver: 4   tos: 0
 len: 106  id: 0x8502 fragoff: 0   flags: 0x2 ttl: 32  prot: TCP(6)
 xsum: 0x7f7c
TCP:             1029 -> netb-ses(139)  seq: 00ba651f  ack: 001351cf
 win: 7333   hl: 5   xsum: 0xd729 urg: 0      flags: <ACK><PUSH>
SMB:  'TreeConX'  retcode: 0x0   retclass: 0x0   AH: 0x0
 netpath id: 0x0000   pid: 0x871a   user id: 0x0000   mutiplex id: 26
 parmcnt(in word): 4   smb_buflen: 4864
```

Fig.3: Example of SMB packet

4. CONCLUSION

Traditional network file systems were created in a world where the dominant network paradigm was the LAN/WAN. Lack of standardization caused that common standard has not been selected and approved yet, despite to ISO standard FTAM (File Transfer and Access Protocol) and recent Microsoft activities with its proposed solution -CIFS.

In addition the network file system challenge has become the distributed file system and connections via Internet/WAN, as there is move from self-contained LAN environments to a world of distributed system and even rarely connected devices. This kind of network environment introduces at least three main challenges:

❶ Authentication
❷ Data Transport
❸ Synchronization

Some of these issues could be solved at the application level rather than the file system level. For instance, Rsync, a powerful program that came out of the Samba project provides remote file synchronization over the network. Another alternative to traditional protocol is for example, **AFS** (Andrew File System) and **CODA**. Coda is a distributed file system developed from AFS2. Coda is designed for mobile computing in an occasionally connected environment. Coda clients maintain a persistent cache of copies of files from the server. Coda checks the network both for the availability of connections between client and server, and for the approximate bandwidth of the connection.

As explained there are still the different solutions in place, such as NFS in Unix and CIFS/SMB in Microsoft world. The future development must consider not only requirement for common file system and protocol, but also to include the new requirements regarding distributed systems, replications, caching, which have not been taken into consideration yet.

Therefore the next development will have to solve the mentioned challanges and to come with a proposal for new solution, which will enable implementation of distributed file system for open and embedded systems as well.

REFERENCES

[1] FRANEKOVÁ, M., KÁLLAY, F., PENIAK, P. VESTENICKÝ, P.: Komunikačná bezpečnosť priemyselných sietí. ŽU Žilina, 2007, ISBN 978-80-8070-715-6.
[2] KÁLLAY, F., PENIAK, P.: Počítačové sitě LAN/MAN/WAN a jejich aplikace, Grada, 2003, ISBN 80-247-0545-1.
[3] PUŽMANOVÁ, R.: Moderní komunikační sítě os A do Z, Computer press 2002, ISBN 80-7226-098-7.
[4] firm documentation