Chapter 12
Session Multiplexor Design Specification

Abstract

Client32 offers session protocol independence. Session protocol independence enables Client32 file and print services to function on any session protocol (e.g., NCP or SMB) that follows this specification.

Introduction

Previous requesters supported only one session protocol: NCP. The Client32 Requester can support any session protocol, and the key to this independence is the Client32 Session Multiplexor, SessMux.

Session protocol providers (e.g., NCP.NLM, SMB.NLM) register their services with the Session Multiplexor. ConnMan then creates, destroys, and maintains connections to remote servers over whichever session protocol the user has chosen.

Figure 1 shows SessMux and the other Client32 NLMs.

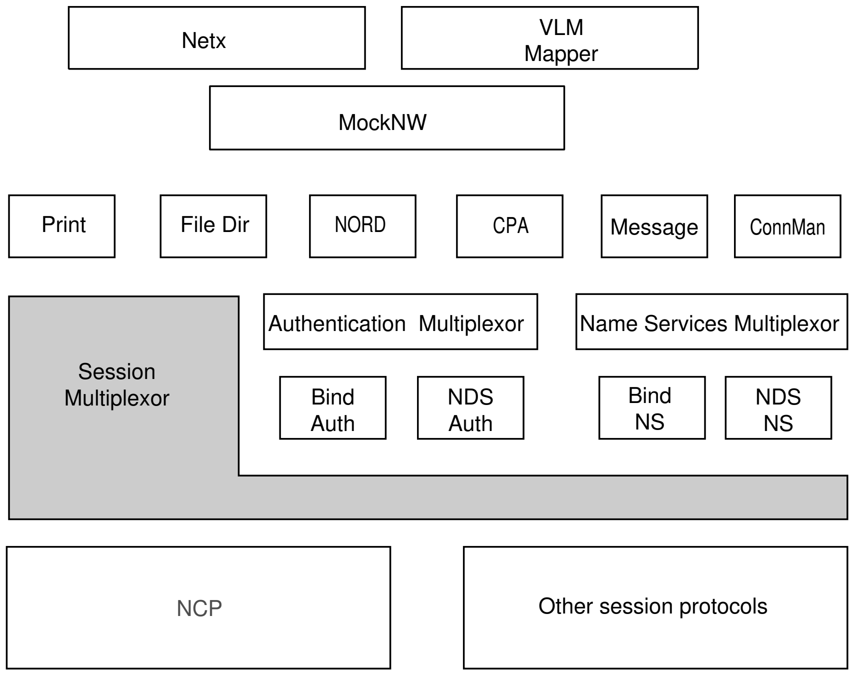

Figure 1. The SessMux NLM and other Client32 Requester NLMs.

Design Description

SessMux is divided into two areas of functionality: registration and multiplexing.

Registration

All available session protocol modules (e.g., NCP) register their services with the session multiplexor. The functions to do this are **SESSRegisterSvc** and **SESSUnregisterSvc**.

The following is the complete set of registry session APIs that have been defined:

SESSRegisterSvc	Registers a session protocol provider with SessMux.
SESSUnregisterSvc	Unregisters a session protocol provider with SessMux..
SESSEnumerateSvc	Lists all currently registered session protocol modules.

All functions to SessMux must pass a connection handle either directly or indirectly so that SessMux can properly redirect the request to the correct session protocol module for processing. SessMux will never enumerate through registered session protocol modules to complete a request (i.e. wildcards not allowed).

Figure 2 shows that SessMux maintains an array of pointers to functions for each of the session protocols that it supports.
When a session protocol provider registers with SessMux, it passes a pointer to these functions. These functions are the session services that each session protocol must implement.

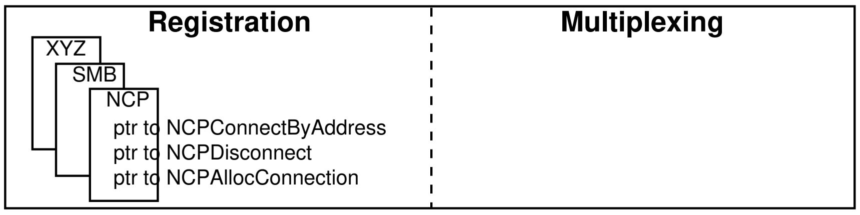

Figure 2. SessMux has two functional areas, registration and multiplexing. This diagram shows that part of the registration function includes maintaining a structure of pointers to protocol-specific funtions.