Universal Data Transporter for NETWARE
Software Development Kit

Presenting the product.

1.

The Client/Server SDK for NETWARE 3.x & 4.x.

This software development kit was designed for those developers who need to write a client/server NETWARE based application.

The Universal Data Transporter SDK was designed to meet all the needs requiered by such client/server applications. The developer will be involved in writing only the request processing modules. These modules will be implemented upon both client and server machines.

1.1. Presenting the layers.

The architecture of a client/server application involves networking considerations. This architecture can be presented as a set of stacked up layers. Figure 1 shows these layers considering the Universal Data Transporter services.

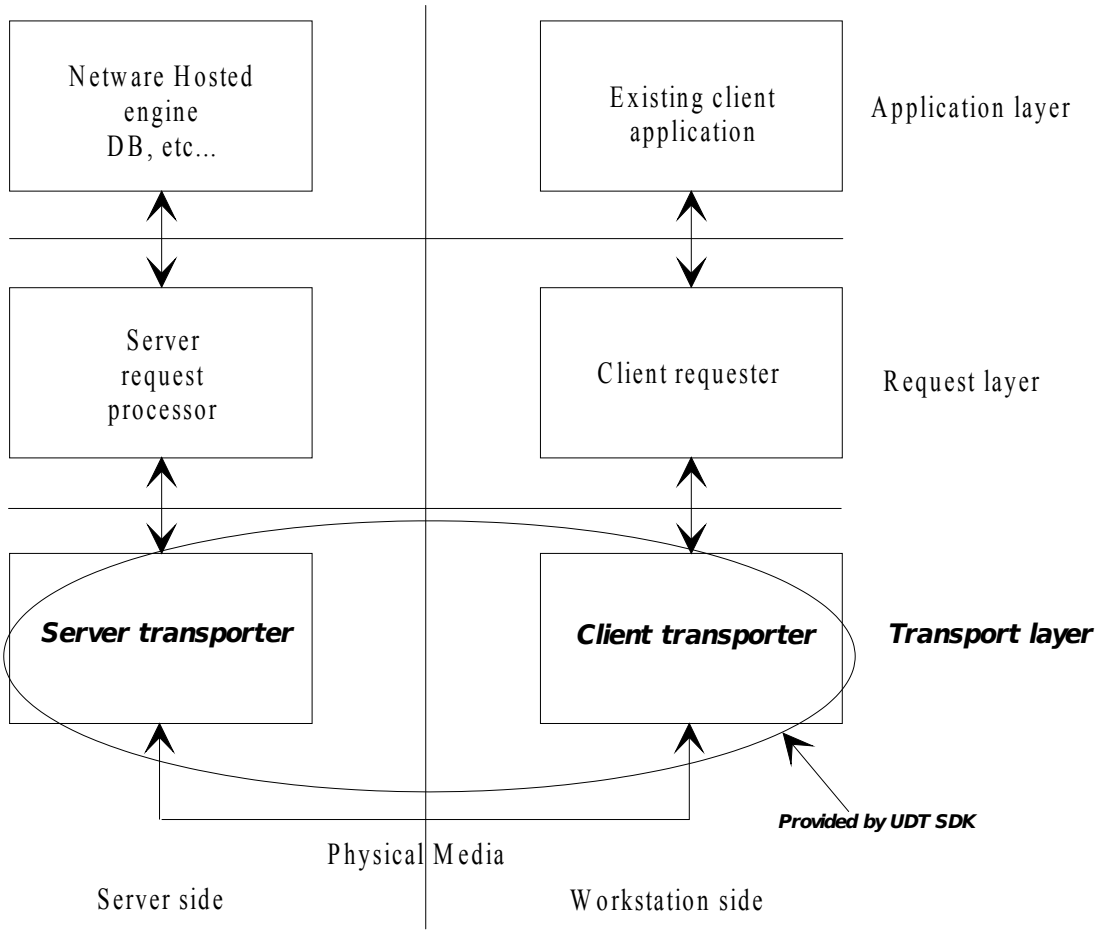

Figure 1.

Considering the application layer as the existing application, the two underneath layers are now needed to implement the client/server architecture.

The request layer will use a software-writer specific format. The layer will need formatting and unformatting the requests. **The Universal Data Transporter SDK gives the developer all the tools needed to implement this layer.**

The most critical layer is, of course, the transport one. As it involves in system specific resources, the development is a long effort. Also, tuning the performances is very difficult as it takes many tests. **This is the layer provided par the Universal Data Transporter.**

That's why this SDK provides an API definition, as callback functions for writing the request processing module on the server and direct calls on to the local Windows/MsDos station. Of course, the server request processor will be implemented in the form of a Netware Loadable Module (NLM).

The client local request processor can be implemented as a WINDOWS DLL or a DOS Lib file.

The two request processing modules (client and server) must implement a unique request format. These modules ignore, though, how these requests flow on the network. Each request must be formatted in a single buffer. This SDK gives you the proper tools to transmit these buffers from the client to the server and back, all transparently!

The SDK contains the following components:

⇒ A Netware NLM as the server request transporter.
⇒ A Windows DLL and DOS static library as the client transporter.
⇒ Sources for the server request processor squeleton (implementing all the call-back functions with proper syntax ready to compile) with a compiled NLM version.
⇒ Documentation for all the public APIs.

1.2. The server transporter.

The main job for the server transporter is to stay running idle on the server machine, waiting for client stations to request connecting to it. At the same time the server supports managing requests receipt for all the clients simultaneously as well as sending back the results to the client. This concurrent task can be done by using the threads feature offered by the 32 bit NETWARE operating system.

The server transporter shall notify the server request processor (implemented by you developer) when both a new client arrives and terminates the session so the module can do any client-related initialization and cleanup processing.

The server request transporter will collect all the received requests (from the client) into a linear address space and pass it up to the server request processor for processing. After passing the buffer, the server transporter stays waiting for the request to be fully completed.

Also, the transporter will notify the request processor at load/unload time. At load time, the request processor must give the number of simultaneous connections allowed.

The server transporter software does not use the NETWARE SAP services to make itself known over the network to all the NETWARE servers for ISDN optimizing considerations. Instead, a dynamic object into the NETWARE bindery will be created at program start time. These objects will be removed before finishing the program. The bindery dynamic object guarantees automatic deletion at remote server reboot time if any problem should occur locally.

This module is implemented in the UDTSRV.NLM file.

Note that the server transporter NLM program can not run stand-alone. It will rely on the presence of a dynamic library as another NLM: the server request processor.

1.3.

The server request processor squeleton.

The request processor will act as a library for the transporter**. This is the module that has to be written by the developer**.

The request processor will be responsible for supporting the following services:

New client session (through a client-based handle provided by the transporter).
Normal client terminate session (through the handle previously given).
Critical client terminate session (through the handle previously given).
Request description text.
Request processing (through the handle previously given).
Respond if loading is possible at transporter load time.
Down itself (or whatever) when the transporter downs.
Give the number of concurrent connections that the transport layer should accept.

These services will be implemented as public API functions and stand as the minimum squeleton for the server request processor implementation. The SDK comes shipped with a software version of this minimum implementation so the developer just has to fill the gaps...

This module must be implemented in the RQPROC.NLM file. The module is automatically loaded by UDTSRV.NLM module.

1.4. The client transporter.

The client transporter is responsible for opening/closing session with several servers simultanously, also for delivering the requests to the server transporter and receiving the results. That's why the client transporter will provide the following services:

Opening a session with a specified server.
Closing the session.
Sending a request/receive the result.
List all the available servers.

These services are implemented as public API functions and will be called by you developper from your client requester software.

This module is implemented in the UDTWIN.DLL (for Windows) or UDTDOS.LIB file (for DOS).

1.5.

The client requester.

The client requester is not part of the SDK. Instead the developer must code the request formatting/unformatting procedure inside it. The formatteded request is then passed to the client transporter that will send it to the server transporter.

This formatted request will be received as-is by the server request processor that will unformat it for processing. It will format the results back to the client so these results will be received back by the client requester. This formatting/unformatting rule remains the responsability of the developer.

1.6. The SDK development environment.

The developer needs the following components in order to write the server request processing modules:

A 486 based machine with 8mb of memory,
The Novell NLM SDK version 3.0 and above,
The Watcom C compiler version 9.0 and above.

The developer needs the following components in order to write the client requester modules:

A 486 based machine with 8mb of memory,
The Microsoft SDK version 3.1 and above,
The Microsoft C compiler version 7.0 and above,
The Microsoft Windows operating system version 3.1 and above.

1.7. Demonstration availability.

A demonstration is available thru José Torres, Fax 33-1-43 32 65 89 or directly on Compuserve at the forum NOVUSER along with the file UDTDEM.ZIP.