

Kernel Mode Driver to Support InterProcess Communication (IPC)

The system of interprocess communication (further IPC) is presented as a user mode library and a kernel mode driver. The driver is a NT-style driver. It is not a device driver. The driver is using public functions of the NT kernel, so it works on all Windows NT platforms starting from Windows 2000.

Its basic purpose is to create a kind of a «bridge» between different processes. The main advantage of the driver as a part of IPC is that it allows exchanging messages unnoticeably for the OS. As it is known, the basis for interprocess communication in different widely known libraries is either the OS windows messages or use of net sockets or shared memory (which is problematic in Windows Vista). All these methods can be controlled by the OS and be blocked on the user level. For instance, at an attempt of using IPC on the basis of sockets appearance of a dialogue, containing permission for the net to be used by the application using sockets, is possible. Messages of windows, starting from Windows Vista, may not reach the addressee at all, though the performance result will be returned as «successfully completed». It occurs due to the difference of «level» privileges for the application working under one and the same account. Using the shared memory becomes problematic at an attempt of exchange between applications working in different sessions. Basing on the disadvantages described above, it was decided to work out a completely independent system. It was proposed to take the code performing high-speed, low-level operations out to driver code, and also to put the high-level logics to the user mode library.

Logically, the driver code can be divided into several logical parts, each of which works with different «pseudo-objects». The basic objects for manipulation are:

- The message object
- The process object
- The global addressee object

The environment (a set of API) as runtime C++ has been created for manipulating the objects. For accelerating the code work, the memory used for the objects is allotted at the system initialization. The disadvantage of this method is a limited number of simultaneously existing objects, but this disadvantage is quite compensated by the speed of work. Allocating the memory solves at once the eternal problem of «leaks».

The memory allocated at the start is also relatively divided into the following parts:

- The memory for the objects itself
- The memory for channel exchange
- The memory for temporary storage of data (a set of slots)

Thus, the relative scheme of IPC driver can be presented as follows:

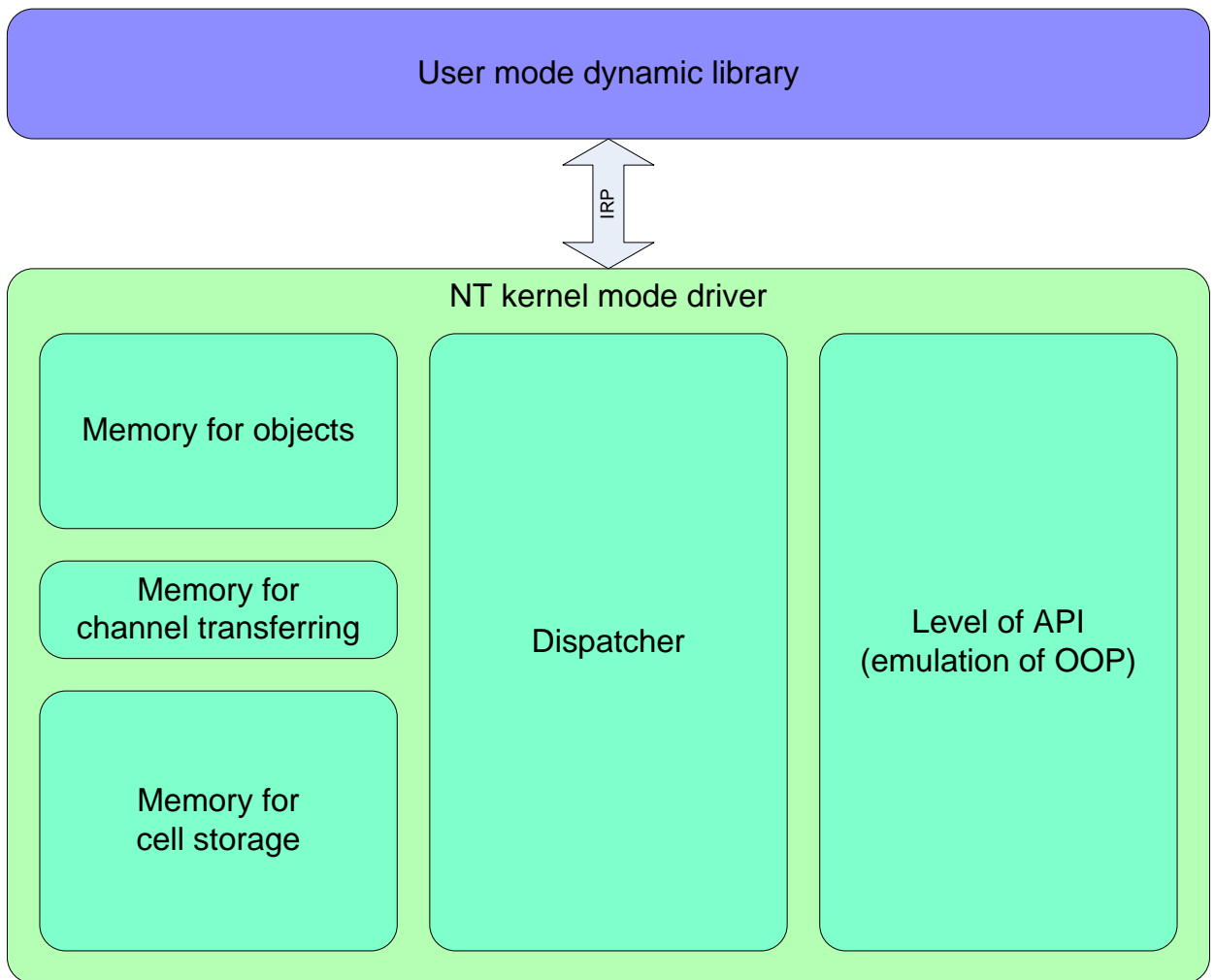


Fig. The general IPC scheme

Objects

Message object is presented as a basic set of data – message code, the first parameter, the second parameter (like a window message), destination address, sender address(probably fictitious), message type (synchronous/asynchronous), etc.

Process object is presented as a set of data describing the current process of the OS in IPC system; a unique identifier of the IPC address of the process, synchronizing objects, messages queue, etc.

Global address object is a special structure storing the description of the IPC address, unique within the present OS. It should be mentioned that if there is a global address, then there is a local address within the current process. Thus, summing up, two addressing directions can be stated: vertical one – within the process, and horizontal – within the system:

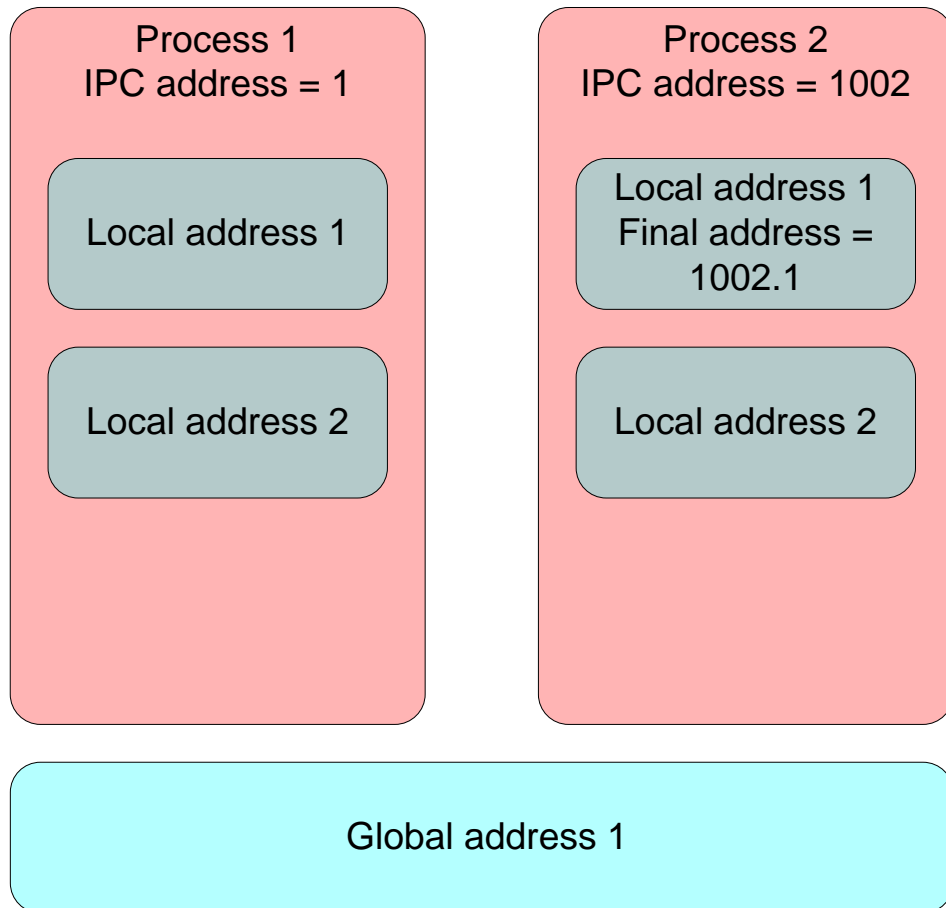


Fig. IPC readdressing

Thus, for forming the destination address, it is necessary to set the type of addressing mode (global or not), the process address and the address inside the process.

The basic messages don't allow sending large amount of data (only two parameters). So, it was decided to realize the data exchange on the basis of messages. The exchange is carried out through shared memory, which is located in the driver. The data itself is compiled in the so called channel containers, i.e. you reserve the channel for passing the data block, pass the data into it, send a message to the handler, then empty the channel, etc. But this method may lead to interlocks, so it should be used carefully.

There is an easier method of data transfer, still containing a restriction by its volume (about 500 kb per one transaction). This method uses the slots memory. This mechanism is something like human post system. The addresser puts the data into a free slot and stipulates the interval during which this data should be stored in the slot. As soon as the interval is out, the data is deleted by the garbage cleaner thread. Such organization allows providing the workability of the system in case of a critical termination of the addressee or addresser work. For example you put the data into the slot, stipulated a 5 seconds' interval, but suddenly your process is terminated and you fail to take the data out of the slot - don't worry, as the garbage cleaner thread will do it for you in 5 seconds.

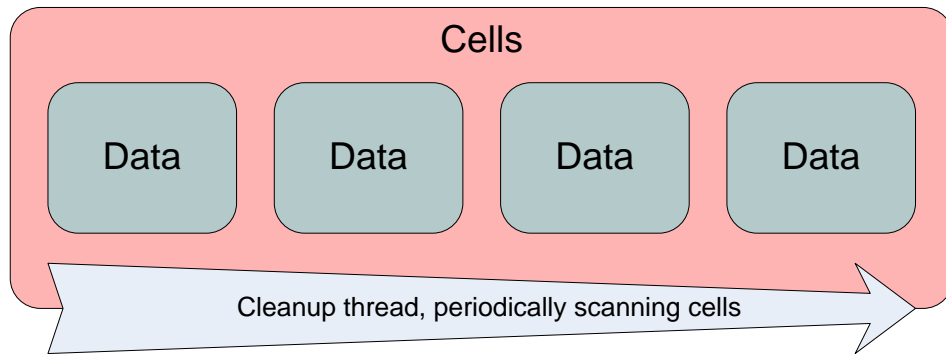


Fig. The system of temporary storage slots

Sometimes, there is a need for sending a message to several addressees at once. For this purpose, a broadcast mailing is foreseen in IPC system. In this case one should only stipulate a local destination address and the message will be delivered to all processes, registered in IPC system.

Organizing the work of a driver is based on messages queues. If there is a message to be handled, it is associated with the process object, which, in its turn, is used by the handling current of the user mode; it is scanning the queues and looking for the messages. The already handled messages are either deleted from the queue (if they are asynchronous), or marked as ready or as once that haven't reached the addressee (if the destination process is no longer existing).

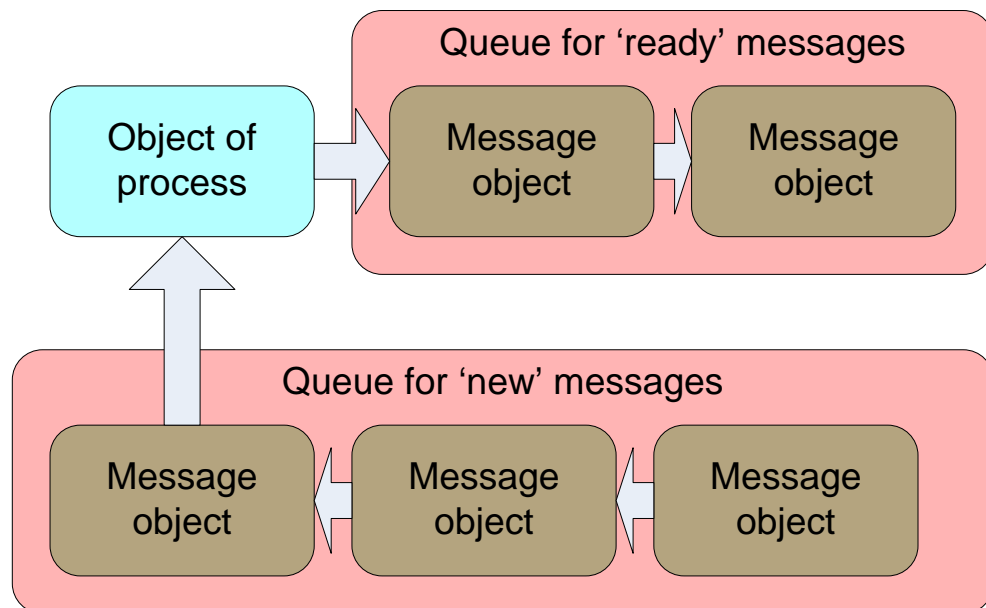


Fig. Messages handling

Settings Storage System Driver

The settings storage system works as the OS registry. It is intended for storing the elementary-type data:

- 1,2,4,8-byte figures
- String and multi-string values
- Binary data

The driver realizes the storing of a tree-like list of names and paths, and also the simultaneous flushing of data to the hard disc. Such scheme is also used on the OS. The difference is that our realization doesn't use the so called system 'handles'. For working with the value you should only stipulate the path to the value and its type. Optimization of the driver work is reached by using the already available memory for storing a new value provided its size doesn't exceed the amount of already allocated block. At reaching the memory limit, there occurs the defragmentation of the whole registry with allotting a block of memory, which is bigger by a definite value.

The speed of search for the needed value is reached by using the binary search algorithm.

For supporting encoding, there is a possibility of installing the packing «at once». The key itself can be passed as a component of the whole name of the element.

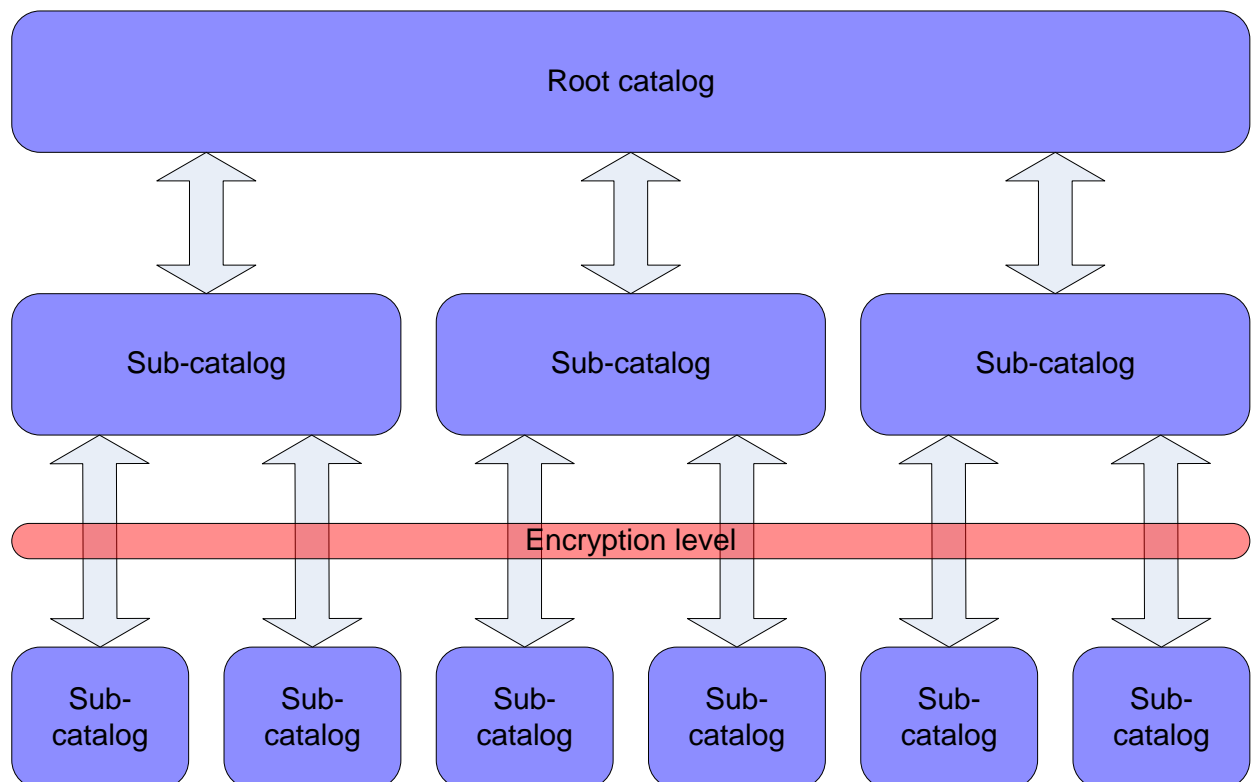


Fig. Settings storage system driver architecture

Network Redirector Driver

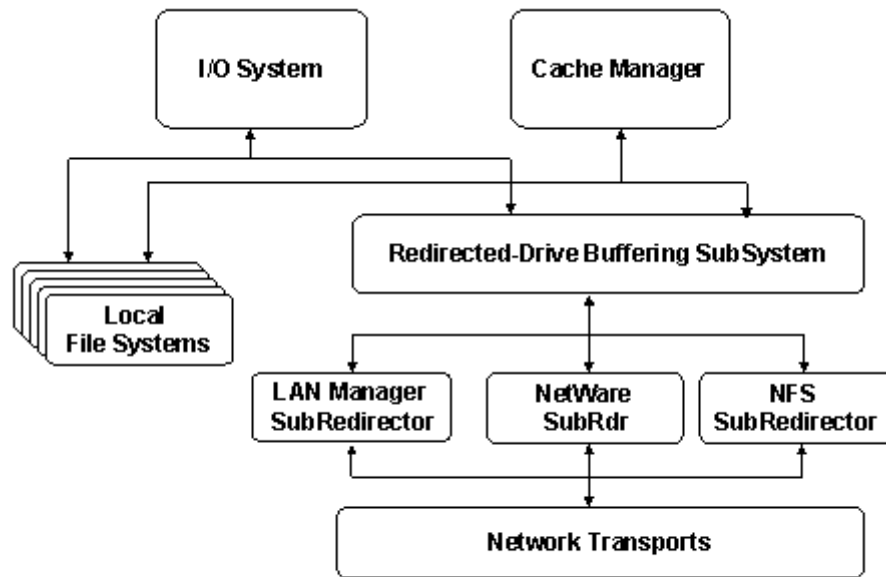


Fig. Mini-redirector architecture in Windows 2000.

Network Redirector – consists of program components installed on the client’s computer used for providing the user the access to the files or other resources (e.g. printers) on the remote system. Network Redirector sends (or readdresses) local applications requests for operations on files to the remote server, where these requests are handled. Network Redirector receives replies from a remote server, which are then returned to the local applications. Network Redirector presents remote files and resources as local ones, on the client’s system, and it also allows to use them in the same way.

Network Redirector is trying to make the access to the remote resources so transparent as it is possible for a local application-client.

A new model of a driver (often called a mini-redirector architecture, or rdr2) for the Network Redirector systems was presented in Windows 2000. This big block of the code was taken out and made public for all Network Redirector drivers. This allows to avoid repeated realization of a complex code for buffering and interaction of the cash manager with the input/output manager.

This public code is called Redirected Drive Buffering SubSystem (RDBSS).

A driver of mini-redirector for providing the user’s access to FTP and WebDav resources has been designed by us. The driver was designed according to all requirements and recommendations to the architecture of a mini-redirector, thus being compatible to the following versions of Windows: Windows 2000/XP/2003/Vista.

File system driver-filter

Driver-filter of the file system is an additional driver adding functions or changing behavior of the file system.

Driver-filter of the file system can filter the input-output operations for one or more file systems or volumes. Depending on the nature of the driver the filter can journalize, observe, change, or even interrupt file operations. The typical applications using driver-filter of the file system are the antivirus systems, encoding programs, hierarchical systems of controlling the disc space.

A driver for monitoring the disc activity and statistics gathering has been designed by us.