NBD NOTES

Week 1 28/06/2023

Thodoris Pontzouktzidis

Network Block Device: Forwarding block device in linux client-server (Standard protocol) TCP

**client <-> network <-> server**

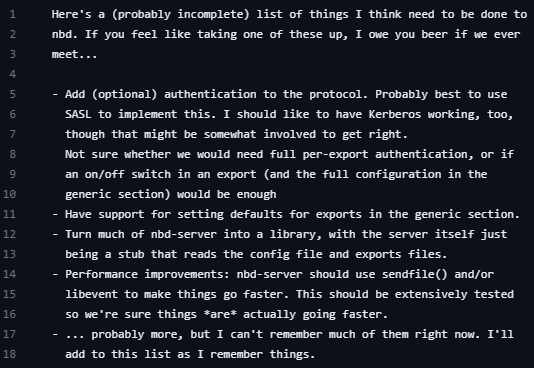
# **Please keep in mind:**

**Reading through:**

1. [**https://github.com/NetworkBlockDevice/nbd/blob/master/doc/proto.md**](https://github.com/NetworkBlockDevice/nbd/blob/master/doc/proto.md)

**This directory contains developer documentation. It's probably not useful unless you wish to help with implementing nbd.**

**This file tries to document the NBD protocol as it is currently implemented in the Linux kernel and in the reference implementation. The purpose of this file is to allow people to understand the protocol without having to read the code. However, the description above does not come with any form of warranty; while every effort has been taken to avoid them, mistakes are possible.**

****

1. [**https://github.com/NetworkBlockDevice/nbd**](https://github.com/NetworkBlockDevice/nbd) **README FILE**

**WILL HELP US SETUP AND RUN WITH NBD.**

1. [**https://www.youtube.com/watch?v=PMa6KFX9AxM**](https://www.youtube.com/watch?v=PMa6KFX9AxM) **This video can also help understand the protocol.**

**Start of proto.md file**

# **Protocol phases.**

Client kernel (or userspace) driver forwards request to server where it is processed by a userspace program.

2 phases handshake , transmission

## **Handshake.**

A connection is established and an exported NBD device along other protocol parameters are negotiated between the client and the server.(1)

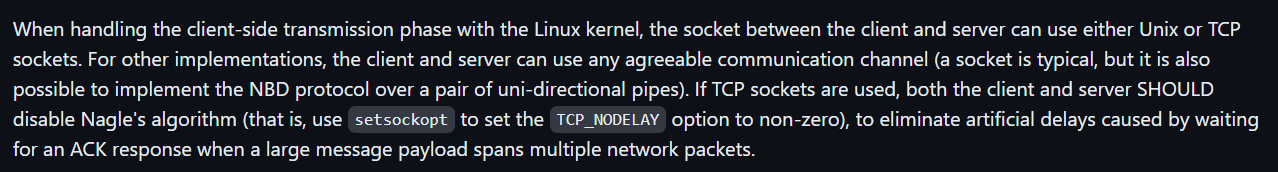
After (1) client performs in **userspace**:

ioctl(nbd, NBD\_SET\_SOCK, sock)

ioctl(nbd, NBD\_DO\_IT)

From Handshake phase to transmission phase

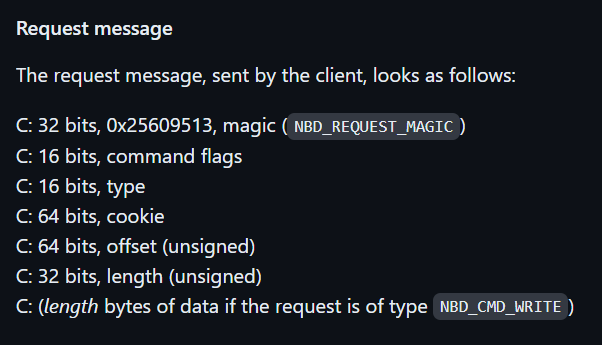
**nbd = file descriptor for an open /dev/nbd0 device node**

****

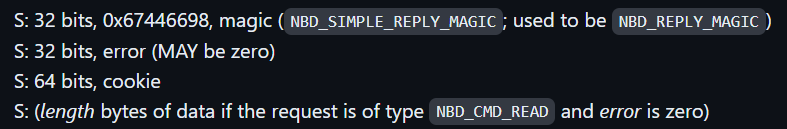
## **Transmission.**

Three message types in the transmission phase: the request, the simple reply, and the structured reply chunk.

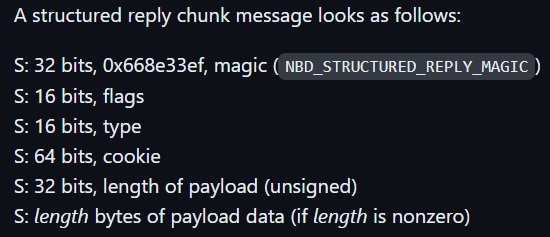
e.g **request**:



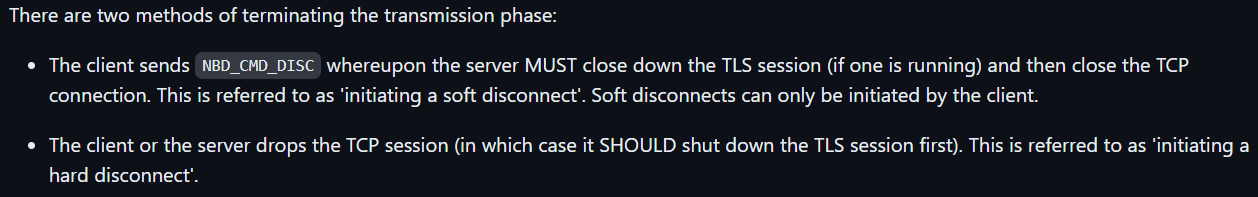
e.g **simple reply** other than NBD\_CMD\_READ and if structured replies have not been negotiated :



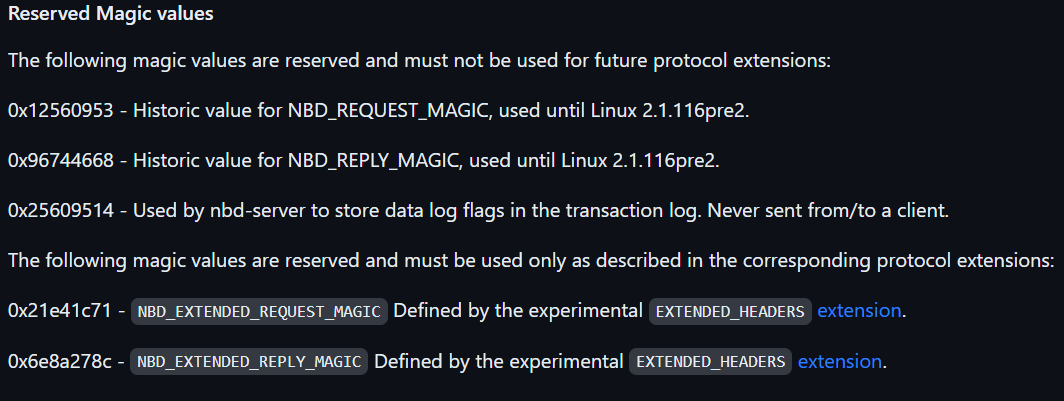
e.g **structured chunk reply :**

****

## **Terminating.**



## **Magic values.**



# **TLS Support.**

# 

Server decides to follow a TLS mode : NOTLS, FORCEDTLS, SELECTIVETLS

Client

# **Size constraints.**

three size constraints: **minimum block, preferred block, and maximum payload**.

**If size constraints have not been advertised or agreed on externally, then a server SHOULD support a default minimum block size of 1, a preferred block size of 2^12 (4,096**)**, and a maximum payload size that is at least 2^25 (33,554,432) (even if the export size is smaller); while a client desiring maximum interoperability SHOULD constrain its requests to a minimum block size of 2^9 (512), and limit NBD\_CMD\_READ and NBD\_CMD\_WRITE commands to a maximum payload size of 2^25 (33,554,432).**

A client MAY choose to operate as if tighter size constraints had been specified (for example, even when the server advertises the default minimum block size of 1, a client may safely use a minimum block size of 2^9 (512)).

**The three size constraints in depth:**

The minimum block size represents the smallest addressable length and alignment within the export, although writing to an area that small may require the server to use a less-efficient read-modify-write action. If advertised, this value MUST be a power of 2, MUST NOT be larger than 2^16 (65,536), and MAY be as small as 1 for an export backed by a regular file, although the values of 2^9 (512) or 2^12 (4,096) are more typical for an export backed by a block device. If a server advertises a minimum block size, the advertised export size SHOULD be an integer multiple of that block size, since otherwise, the client would be unable to access the final few bytes of the export.

The preferred block size represents the minimum size at which aligned requests will have efficient I/O, avoiding behavior such as read-modify-write. If advertised, this MUST be a power of 2 at least as large as the maximum of the minimum block size and 2^9 (512), although larger values (such as 4,096, or even the minimum granularity of a hole) are more typical. The preferred block size MAY be larger than the export size, in which case the client is unable to utilize the preferred block size for that export. The server MAY advertise an export size that is not an integer multiple of the preferred block size.

The maximum payload size represents the maximum payload length that the server is willing to handle in one request from the client. If advertised, it MAY be something other than a power of 2, but MUST be at least as large as the preferred block size, and SHOULD be at least 2^20 (1,048,576) if the export is that large. Advertising a maximum payload size of 0xffffffff is permitted when the server does not have a fixed limit on client request payloads. Typically, the advertised maximum payload length is independent of the export size, even though the actual payloads for read and write cannot successfully exceed the constraints given by the export size and offset of a request. Notwithstanding any maximum payload size advertised, either the server or the client MAY initiate a hard disconnect if a payload length of either a request or a reply would be large enough to be deemed a denial of service attack; however, for maximum portability, any payload not exceeding 2^25 (33,554,432) bytes SHOULD NOT be considered a denial of service attack, even if that length is larger than the advertised maximum payload size.

# **Metadata querying.**

Metadata support is present by client queries ( Metadata querying section at <https://github.com/NetworkBlockDevice/nbd/blob/master/doc/proto.md#metadata-querying> ).

May need this later.

# **Values.**

This section describes the value and meaning of constants (other than magic numbers) in the protocol.

When flags fields are specified, they are numbered in network byte order.

The link covers VALUES that have to do with: **Handshake phase, Transmission phase, Experimental extensions.**

[**https://github.com/NetworkBlockDevice/nbd/blob/master/doc/proto.md#values**](https://github.com/NetworkBlockDevice/nbd/blob/master/doc/proto.md#values)

# **Compatibility and interoperability.**

Originally, the NBD protocol was a fairly simple protocol with few options. While the basic protocol is still reasonably simple, a growing number of extensions has been implemented that may make the protocol description seem overwhelming at first.

**This section of the proto file divides NBD in “optionals” and “musts” for people who want to implement the Protocol.**

# **Future considerations.**

Structured replies; the Linux kernel currently does not yet implement them.

**End of proto.md file**

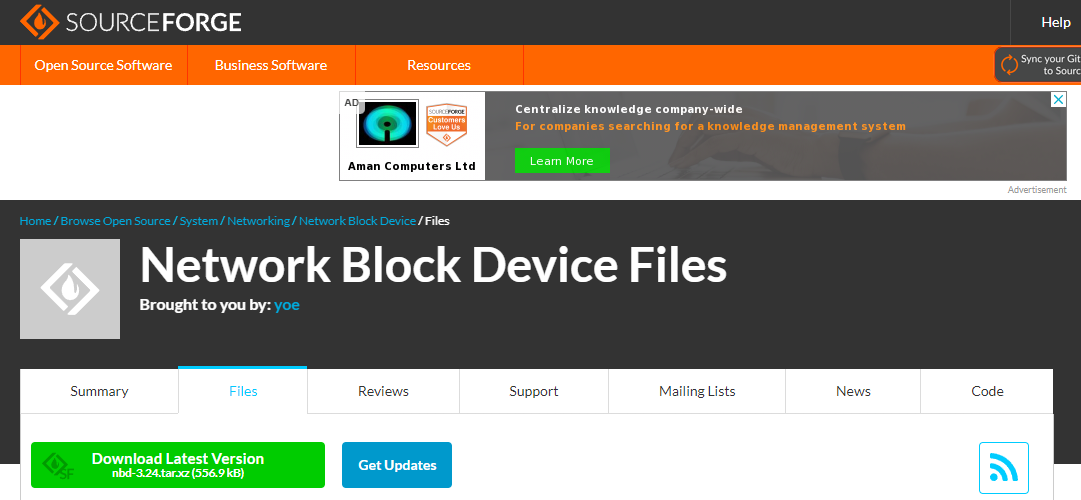
**======================================================================**

**Start of README.md file**

## **NBD README.**

This <https://github.com/NetworkBlockDevice/nbd> repo(package) contains nbd-server and nbd-client.

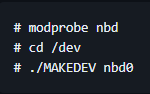
**To install the package, download the source and do the normal configure/make/make install dance. You'll need to install it on both the client and the server. Note that released nbd tarballs are found on** [**sourceforge**](http://sourceforge.net/projects/nbd/files/nbd/)**.**



**For compiling from git, do a checkout, install the SGML tools (docbook2man), and then run './autogen.sh' while inside your checkout. Then, see above.**

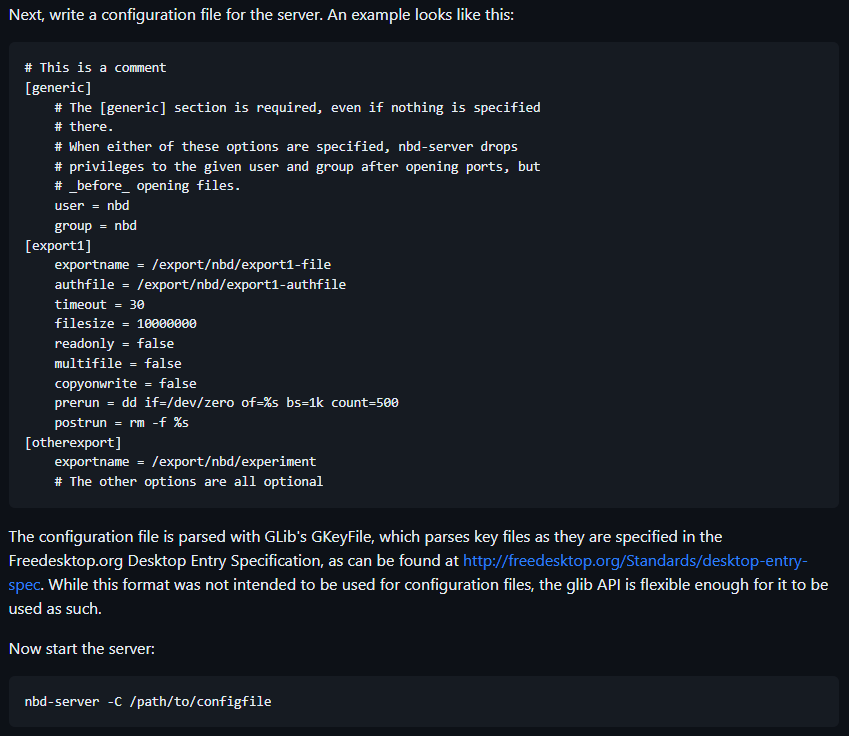
## **Using NBD.**

First, on the client, you need to load the module and, if you're not using **udev**, to create the device nodes:



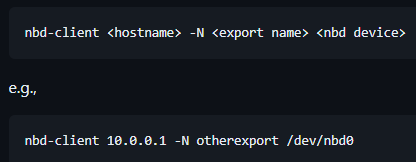
(if you need more than one NBD device, repeat the above command for nbd1, nbd2, ...).

To start the **server.**



**The path is absolute.**

To start the **client.**

****

**nbd-client** must be ran as **root**; the same is **not true** for **nbd-server.** (but do make sure that /var/run is writeable by the server that nbd-server runs as; otherwise, you won't get a PID file, though the server will keep running).

<https://github.com/NetworkBlockDevice/nbd#badges> here we can see the available packages for each corresponding OS

**End of README.md file**

**======================================================================**

**Running TeraHeap with NBD**

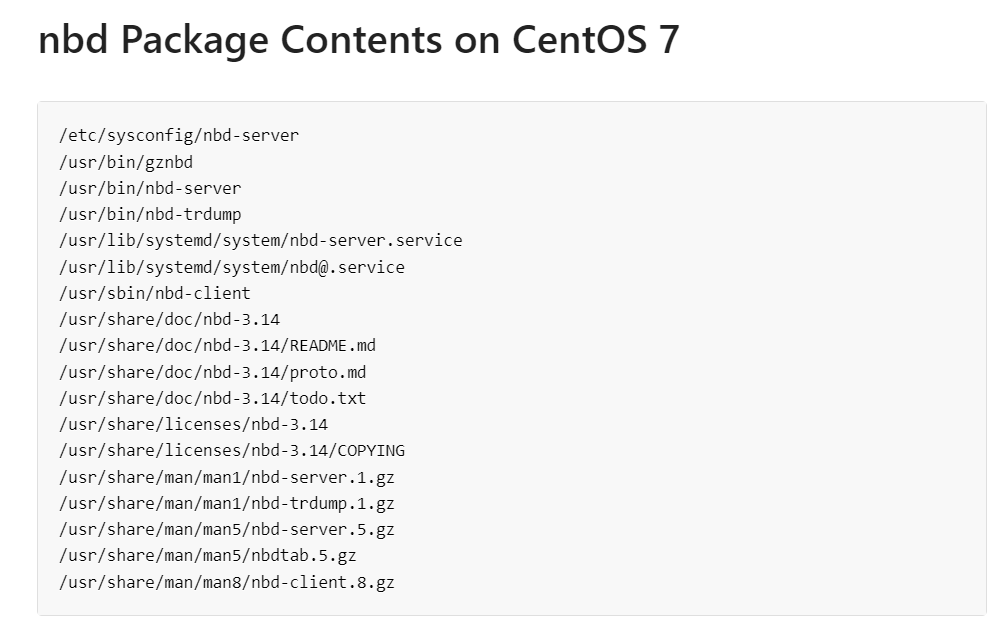
## **Setup.**

A cool setup way (on Ubuntu): <https://sweetcode.io/introduction-to-linux-network-block-devices/>

**For Centos ‘?’ we will use yum to install nbd :**

* **sudo yum makecache**
* **sudo yum -y install nbd**

NBD package contents:

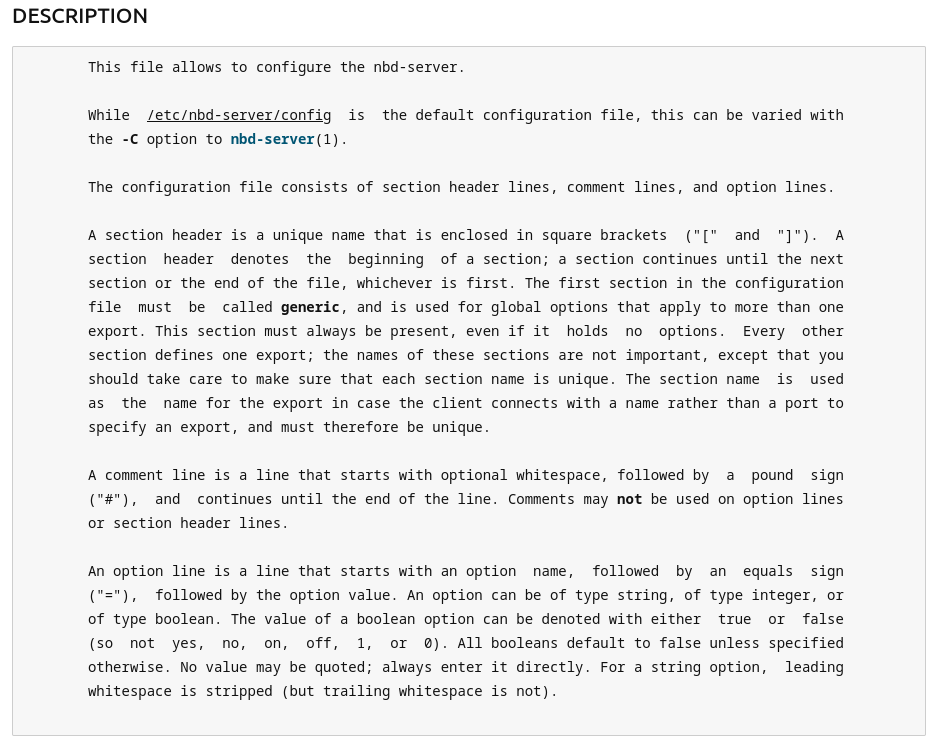


After this we can “Probably” use NBD like any other distro…

**Let’s try NBD between a ubuntu vm(server) and a kali (client). I will create a nbd client-server relation and write a simple c file that does some basic file operations.**

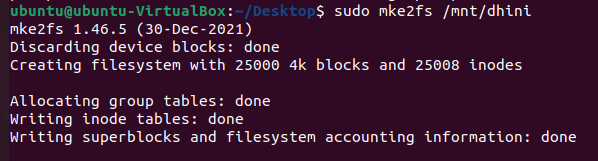
**We will connect to vm using ssh after that we will connect to the nbd-server and export the block device to the kali host.**

**to write a configuration file you can use this man**

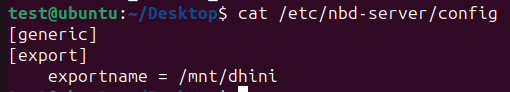
****

## server

* **apt-get install nbd-server** for the server (ubuntu)
* **dd if=/dev/zero of=/mnt/dhini bs=1024 count=100000** to create a file of 1.024(bytes) **\*** 100.000=102.400.000 bytes ~102MB
* **mke2fs /mnt/dhini** to make it as a filesystem



* **write a config file for the server**

****

**under /etc/nbd-server/config**

* **run nbd-server the default port is 10809/tcp**
* **confirm that the process is running with ‘htop’** (also the only way i found to stop the server is with pkill …)





## client

* **apt-get install nbd-client** for the client (kali)
* **sudo nbd-client <server-ip> -N Mu /mnt/dhini** (because i run server on vm with NAT i cant access the vm from the host that means that the tcp socket can not be made **Error: Socket failed: Connection refused.** I need to port forward from the the host to the vm but i can't seem to get it right **this will be no issue when we run this on a real network e.g the cluster**)

**followed steps from official documentation:**

**server**

[**https://manpages.ubuntu.com/manpages/focal/man1/nbd-server.1.html**](https://manpages.ubuntu.com/manpages/focal/man1/nbd-server.1.html)

**client**

[**https://manpages.debian.org/testing/nbd-client/nbd-client.8.en.html**](https://manpages.debian.org/testing/nbd-client/nbd-client.8.en.html)

## **IMPORTANT**

**Because NBD uses tcp we will have huge overhead when teraheap tries to access the exported block device. Luckily RNBD exists which uses rdma instead of tcp. We will create a block device on a ram disk and export it to the client (where teraheap is running) . The access speed will be much quicker because of the rdma. It would be wiser to to get teraheap test numbers with RNBD instead of NBD but I guess it is worth it that we got to know nbd which might have similarities with RNBD.**

**First some things to keep in mind about teraheap:**

* teraheap uses a second high-capacity heap (H2) over a fast storage device implemented in **OpenJDK**.