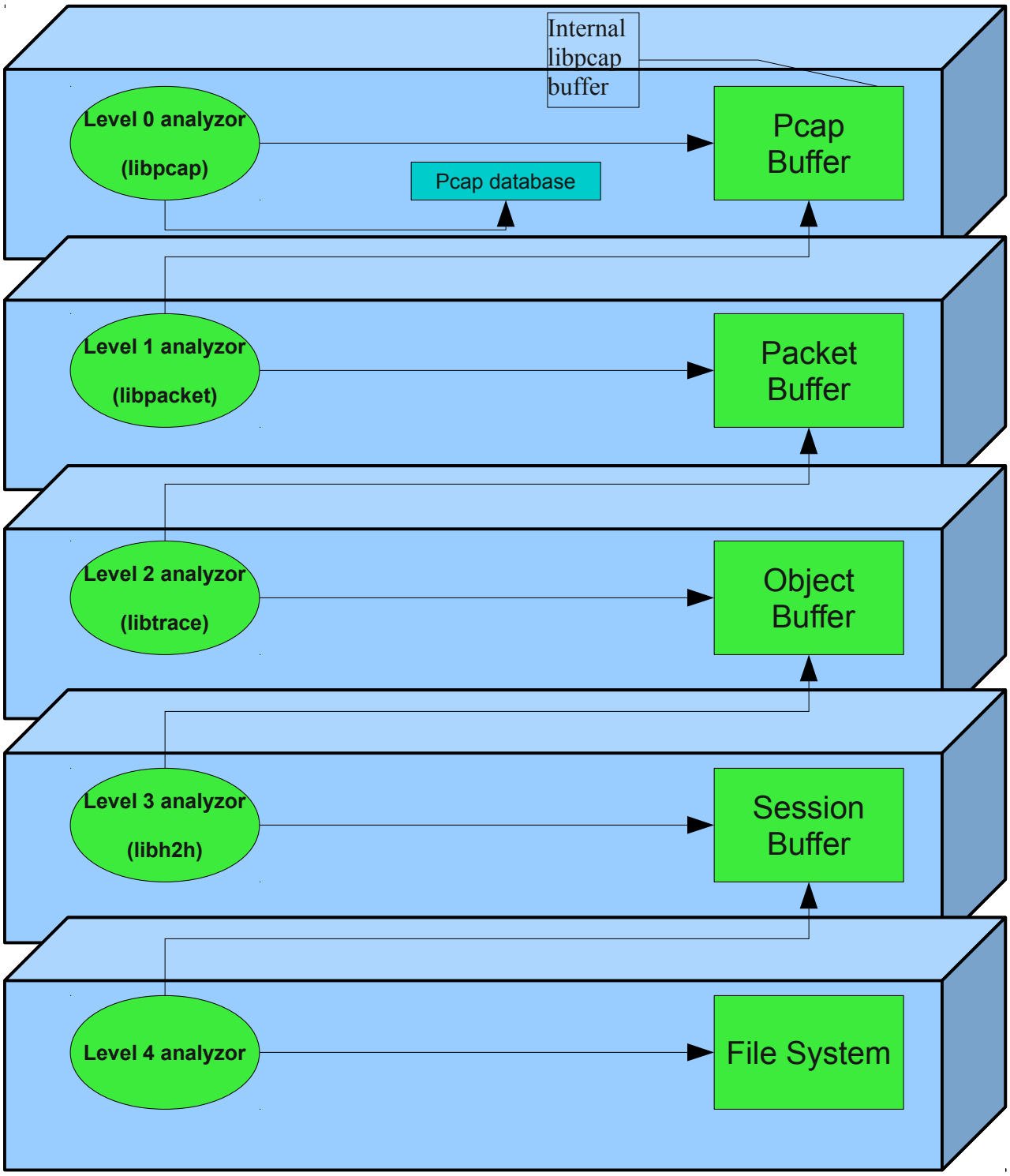
In the name of God

Notes

1. As we know, we are disiging a general perpose sniffer. So, we need to know and understand more than just web and html/http to implement our product. For example, smtp or imap at least in plain mode. The centeral questions are:
   1. Is it possible to capture other related protocols like smtp, imap and others by this structure?
   2. Is this structure can give us complete and comprehensive abstraction model of capturing process for other protocols besides http?
2. While we are focused on layer 7(application layer) in sniffing, we should not suppress lower level protocols. For Example, pptp and other tunneling protocols.



Description

Our design looks like a building from the top. In fallowing sections, we will describe each level with point to some important details. Note that this model, illustrate an abstraction model of core fucntionality not exact implementation model. Of course it needs more details and edition to work in real world.

Level 0:

You see the floors started from 0. Level 0 indicates the main and first start point of program. In other hand, level 0, is Pcap library mechanism and functions. We will use Pcap functions to sniff raw packets crossing through NICs to its destinations and get a snapshot from each one in kernel level. Beside these, we use a very useful and interesting Pcap functionality, named, off-line sniffing to store and save packets into one or more files. This functionality is named "**Pcap database**" it the above illustration and distinguished by blue color.
Pcap library, during setup, will asking us to set size of internal buffer. This show we have an internal bucket. So we display that in the above structure by "**Pcap Buffer**".
This level, will come to end at **"pcap_loop"** function, where sniffed packets from buffer are ready to deliver to next floor or level.

Level 1:

This level, start from fist function fallowing after "**pcap_loop**" function. It means, start point of this level, will be the place, upper level packets are delivered.
In this level, we just take care about fragmented packets to reassemble them and recreate a complete and original packet. Probably we use "**libpacket**" to provide this functionality.
At least, we will create a table,in order to put raw packets in it to provide a repository of incoming packets like a buffer. This buffer named, "**Packet buffer**".

Level 2:

This level start with fetching reassembled packets from the upper level. So we will try to recreate original file or object from these packets. These objects, include: web pages elements like: CSS, JavaScript, images even audio of video files, SMTP or email elements like HTM/HTML mime file types or other elements in messaging protocol like XMPP.
So, will create objects and put them into another table called, "**Object Buffer**".

Level 3:

This level start from fetching objects from upper upper level table. This level, is most important and considerable level in our design.. Actually, our creativity will place into this **level to recover and rebuild real and meaningful information are crossing through the network.**
We should combine recovered objects from upper level to rebuild sessions like a complete web page, a yahoo messenger session with its all sent and received messages, or an email with meaningful information and view.
This will done by our library maybe some already existed like, libhtm2html and others.
This level, will end until reach to a complete response or meaningful information and preparing for store them into appropriate place, based on configurations.

Level 4:

This level starts from fetching response or sessions from upper level to store them in a appropriate method. This methods include: store them in file system, sending them across mail/web services even by encrypting them and so so.

End.