Project Report: A Study of Net Subsystem Evolution in Linux

Saurab Dulal
{sdulal}@memphis.edu
University of Memphis

Abstract – Linux connects with external systems/hardware over the network with the help Network Subsystem. It is one of the most important components of the kernel and is constantly evolving to support billions of devices. In this project, I performed a basic study on Linux Network Subsystem by download and analyzing the patches applied to it over the past 14 years from 2006 to 2019. Altogether, about 132 thousand patches applied to the net subsystem were analyzed. According to statistics, it shows that about 75 thousand patches were applied to the drivers and about 56 thousand patches were applied to the core in that time frame. The patches were categorized and analyzed based on the counts, and some other factors (e.g. year, months, bug fix, feature enhancement, etc). The project repository and collected data can be accessed here [5]

I. Introduction

Linux Net Subsystem is one of the crucial components of the Kernel. It deploys several interfaces, protocols, drivers, etc inside the kernel that facilitates the communication with external device hardware and with several operating system services. The figure 1 shows a basic Linux Network Subsystem architecture, its major interfaces, and protocols along with information in and out of the kernel. The network device driver –implemented at the bottom of the interface communicated with the external driver. It maintains a one-to-one relationship with the hardware device and the driver, whereby maintaining a separate list of drivers for each hardware component. It also provides a consistent view of all the hardware devices to the upper layer so that the upper layer doesn't have to worry about external hardware semantic. Similarly, the network protocol module implements a wide variety of transport protocols. The protocol Independent interface module is used by other subsystems within the operating system to access the internet. Interestingly, it also facilitates the access of the internet independent of protocols or hardware. Finally, the system call interface aids communication with the virtual system and also with the programming interface using system calls. The overall Net subsystem is designed and is evolved in such a way that it has supported thousands of devices having their own implementation semantics and protocols. This seamless compatibility of the Linux network interface with a wide variety of external hardware is achieved through hundreds of changes, patches, bug fixes, and continuous evolution in the Net subsystem. The progress and maturity achieved in

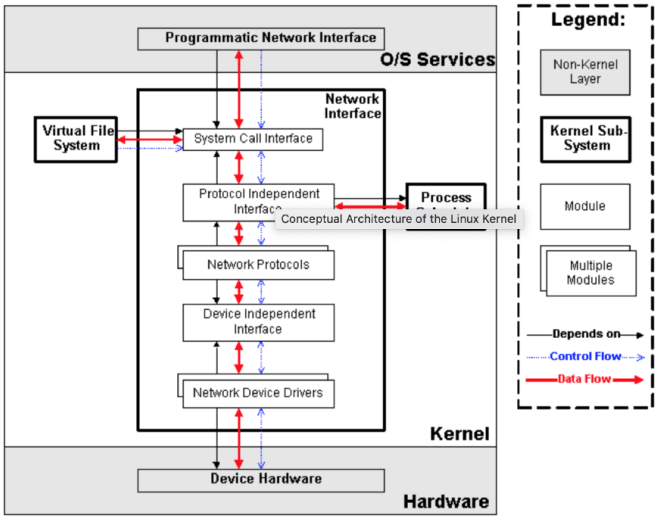


Fig. 1: Network Interface Subsystem

Subsystem code is somewhat a benchmark for the iterative development process, detail study can facilitate software development process for other big system and newly incoming submodules.

Through this project, I did several statistical analysis on the patches submitted for the core net subsystem and to the drivers in the past 14 years by thousands of developers from across the world. Altogether, according to my method of data collection, I found about 132 thousand patches submitted for the net subsystem. It includes the patches submitted for the net drivers as well. I download the Linux kernel source code from GitHub and wrote few scripts to parse the patches. I used mongo DB to store the data and R/python for data analysis. The analysis performed was mostly static – meaning just counting numbers of patches for a different category (which I will explain in the later section). Due to the time limitation, I was not able to do semantic analysis on the patches and have left it for the future work.

II. Methodology

A. Data Collection

First, Linux kernel source code was downloaded from the Github repository [4]. The size of the source code was about 4GB. Next, with some the help of online resources and manual inspection, I found out that most of the patches submitted for the core net subsystem start with submodule $\langle name \rangle$ colon

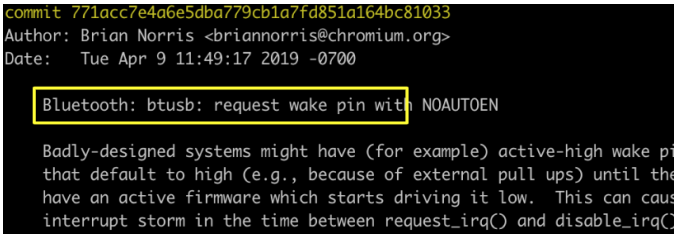


Fig. 2: Git log for a patch submitted for net/Bluetooth

(:) and the rest of the commit message. Submodule names are the directory names found inside **net** folder in the source tree. There are altogether 71 major submodules under the net directory. Some of the examples of submodules are NET, 6LOWPAN, 802 8021Q, 9P, KCONFIG, MAKEFILE, etc. An example of a patch submitted for net subsystem's submodule looks like figure 2. So, I wrote a python method to get all the patches whose commit message starts with any of the $\langle NetSubmoduleName:\rangle$ [ref IV-A] and stored them. Since this method only captured the patches submitted for the core net subsystem and not the driver, I needed a new strategy for parsing patches submitted for net drivers. This was a bit tricky because there was no standard format to submit patches for drivers like in the case of core patches. The patches were submitted with some $\langle name\rangle$ colon (:), but they were not so informative, even if they are then it's on much lower granular level. Also, the names that came with the patches were very hard to track. Finally, I decided that if any patch modifies files inside *drivers/net* directory, I will consider that as a net drivers patch. Note that in this process all the patches with Merge status were ignored to remove the redundancy. To categorize the patches as a bug fix or feature enhancement, I searched for a keyword "fix" in the commit message. And if found, the patch is categorized as bug fix otherwise feature enhancement. Thus in total, 132k patches were collected from Jan 10, 2006, till April 12, 2019, for the core net subsystem and the net drivers.

B. Analysis Metrics

As mentioned earlier, only some of the basic analysis based on the counts was done on the patches such as i) patches submitted per year ii) patches submitted per month (aggregated by year) iii) patches submitted for the core and for the drivers by year iv) patches by bug fix or feature enhancement, and finally v) patches submitted for each submodule of the core net subsystem e.g. patches for 6oLoPAN, UNIX, Bluetooth, IPv6, IPv4, etc.

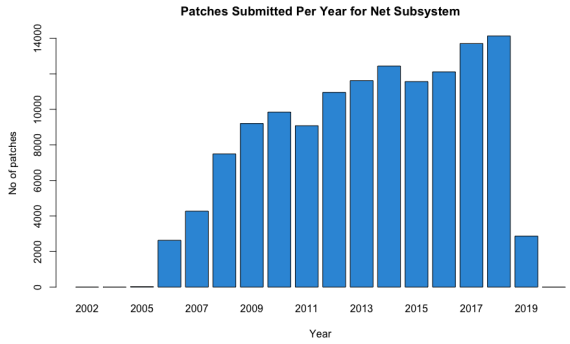
C. Data Processing

First, the hash of every patch was obtained. This was done by parsing git commit message using git log command. A typical git log command contains, commit hash, authors name, date submitted, commit message, etc. Next, using a git command 'git show $\langle hash\rangle$ – stat', all the necessary information of desire was collected. The git stat command provides additional information about a patch such as how many and what files were modified, lines of code added/deleted, who approved the patch, and so on. Processing these pieces of information helped to figure out if a patch was for core net subsystem or for the net drivers. Also, as explained earlier the patches that contain keyword "fix" in the commit message are categorized as bug fix else feature enhancement. Next, a data object was created using the following information.

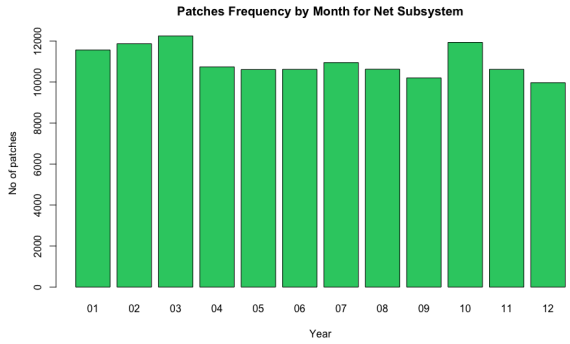
1) CommitID – *Commit hash of a patch*
2) Author – *Author Name*
3) Date – *Committed Date*
4) ModificationStatus – *Files modified, lines added/deleted*
5) SubModuleName – *If core, submodule name*
6) isCore – *Is core or net driver*
7) isBugFix – *Bug fix or feature enhancement, this is done by checking if words such as Fix is present on a patch or not*

In the next step, this data object was stored in a collection table of Mongo DB running on a local environment. Once all the data were collected, a next script, written in R, was used to process the data and create the desired graphs.

III. Results



(a) No of patches per year



(a) No of patches by month

Fig. 4: Patches Submitted on the basis of either year or month (aggregated)

The above graphs show that every year the patches applied to the net subsystem are increasing linearly. About 14k patches were submitted in 2018 alone, which is nearly 10% of the

total patches submitted and compared to 2006, it is more than 600%, this is a huge growth in net subsystem in a span of 14 years. And in 2019, approximately slightly more than 3k patches are already submitted and we can expect thousands more. Since new subsystem such as 6LOWPAN (for IoT devices) are being developed rapidly, we can expect more patches this year compared to last year. The second figure [4a] shows that the frequency of patch submitted each month (aggregated in years) is nearly equivalent. Slightly is grater the beginning of the year.

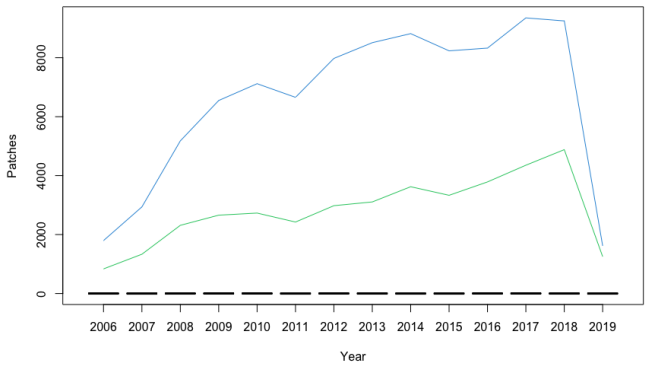


Fig. 5: Bug fixes vs Feature enhancement
Blue line: Feature enhancement
Green Line: Bug fixes

The figure [5] shows that more patches are submitted for bug fixes than for feature enhancement. This is a bit suspicious because in most of the software development environment counts for the bug fixes are usually higher than the feature enhancement. Nonetheless, if this is a true case then it might relate to several features net subsystem develops each year to support billions of devices. In all these years, we have seen that the list is very dynamic – several protocols become obsolete and many new ones come each year. Thus, patches for features might be more than for bugs. However, this needs rigorous study and experiments to come up with actual trends.

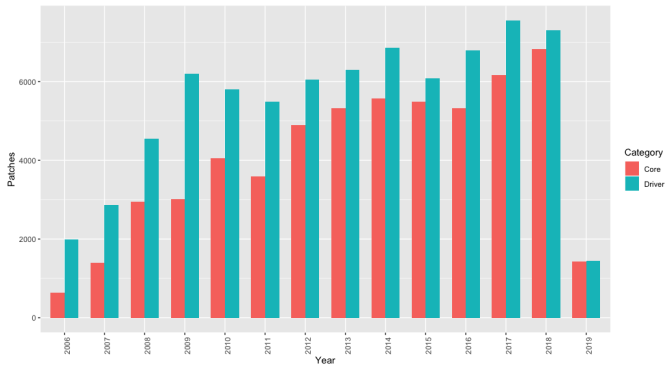


Fig. 6: Patches submitted for Linux core and drivers by year

The next figure 6 shows a bar diagram of patches submitted for the drivers and for the core net subsystem. It shows that almost every year more patches are submitted for drivers than for core. Intuitively this makes sense as well because drivers are usually developed by third-party hardware companies such as Intel, AMD and so on, and can have a high chance of hitting bugs – Linux core is huge, its unfathomable in short span of time. Also, these third parties bring lots of changes to their drivers very frequently in order to incorporate the upgrades and new features in the hardware industry.

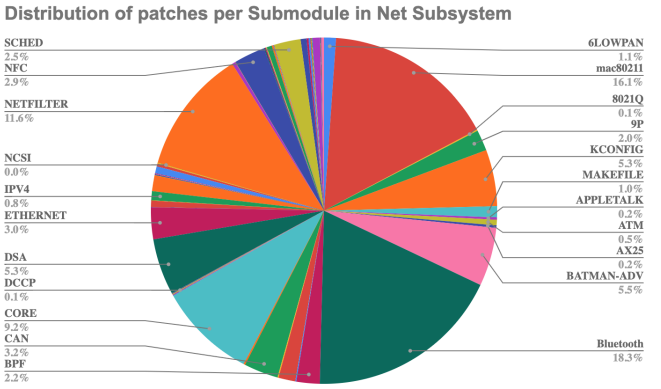


Fig. 7: Patches by submodules under Net Subsystem (drivers not included

Finally, the last figure shows a pie chart of patch distribution across the core net subsystem submodules (this does not include patches submitted for drivers). It shows that the highest no of patches are submitted for Bluetooth ($\langle 18.3\% \rangle$) in the past 14 years. And the second most are submitted for mac80211 – this provides a hint on the rapid progress in the field of communication and compatibility. Some of the new submodules such as 6LoWPAN, recently developed for IOT is getting popular.

IV. Conclusion

In this project, I collected about 132k patches submitted for Linux Net Subsystem in the past 14 years, categorized them in several granularities such as features or bugs, core or driver, and in several net subsystem submodules. And finally, I performed some basic statistic on them based on the year the patches were submitted, and on the previously mentioned features. Due to the time crunch, I could not go into deeper analysis, so several other works can be done in the future with the collected data. Such as i) analysis of the patch on the basis of semantic and find out the similarity between the patches ii) study the bug pattern and categorize the patches as functional or technical, and also study the consequences bugs on the patches iii) study the correlation between the code and the bugs and so on.

Resources: https://github.com/dulalsaurab/net_subsystem

References

[1] Lu, Lanyue, et al. A study of Linux file system evolution. Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 13). 2013.

[2] Andy Chou, Junfeng Yang, Benjamin Chelf, Seth Hallem, and Dawson Engler. An Empirical Study of Operating System Errors. In Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP 01), pages 73-88, Ban, Canada, October 2001.
[3] Nicolas Palix, Gael omas, Suman Saha, Christophe Calves, Julia Lawall, and Gilles Muller. Faults in Linux: Ten Years Later. In Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XV), Newport Beach, California, March 2011. 2
[4] Linus Torvald, Linux kernel source tree, https://github.com/torvalds/linux, accessed on April 16, 2019
[5] Saurab Dulal, Study of Net Subsystem, https://github.com/dulalsaurab/net_subsystem

A. Sub Module Names

Following are all the submodules under net directory of Linux source code.

6lowpan, can, iucv, nfc, strparser, 802, ceph, kcm, nsh, sunrpc, 8021q, key, openvswitch switchdev, 9p, core, l2tp, packet, Kconfig, dcb, l3mdev, phonet, tipc, Makefile, dccp, lapb, psample, tls, appletalk, decnet, llc, qrtr, unix, atm, dns_resolver, mac80211, rds, vmw_vsock, ax25, dsa, mac802154, rfkill, wimax batman-adv ethernet, mpls, rose, wireless bluetooth hsr, ncsi, rxrpc, x25, bpf, ieee802154, netfilter, sched, xdp, bpfilter, ife, netlabel, sctp, xfrm, bridge, ipv4, netlink, smc, caif, ipv6, netrom