Carve your NetBSD

Pierre Pronchery

Freelance IT-Security Consultant
DUEKIN Consulting
<pierre.pronchery@duekin.com>

Guillaume Lasmayous

<guigui2@guigui2.net>

Abstract

After over 20 years of active development [1], NetBSD [2] proves to be a resilient, attractive, featureful and stable platform for industrial products and research projects alike [3]. The reasons behind the technical and practical merits of the system will not be explored or debated here; however, there is always space for improvement. This paper (and associated talk) attempts to identify areas in which gaps may be determined, and presents ways and ongoing work to address them. The topics covered range from the development model to a more user-oriented release strategy, through the adoption of key industrial processes. The EdgeBSD Project is introduced [4] as a platform to experiment with these propositions. Additionally, user interfaces for both desktop and embedded environments are demonstrated, thanks to the DeforaOS Project [5].

[1] Initial revision for `src/Makefile`, *http://cvsweb.netbsd.org/bsdweb.cgi /src/Makefile?rev=1.1&content-type=text/x-cvsweb-markup*

[2] The NetBSD Project, *http://www.netbsd.org/*

[3] « What is NetBSD? » in the NetBSD Guide, *https://www.netbsd.org/docs/guide/en/chap-intro.html*

[4] The EdgeBSD Project, *https://www.edgebsd.org/*

[5] The DeforaOS Project, *https://www.defora.org/*

Introduction

NetBSD is one of the first Open Source projects to have adopted a community-based development model [6]. Established and first released as far back as 1993, it is developed using a "global" approach to the system, with the kernel and essential user-land components integrated together as a whole [7]. This is very different from the approach observed in the omnipresent (GNU/)Linux community, where hundreds of software distributions respectively collect and integrate myriads of components together, as found within the global Open Source Software "bazaar" [8].

The development workflow for NetBSD reflects this desire of a clean, integrated platform for the base system and packages. While generally praised for the quality of the designs and solutions adopted, it can also be harmful for the growth and renewal of its developer base [9]. Moreover, the centralized source code management tool behind development, CVS, no longer fits the distributed approach in increasing use within the industry [10].

The audience met by the NetBSD Project in its current form is largely composed of software developers and experienced users. While a number of software and hardware companies ship products based on NetBSD [11], this is usually not advertised as such, and few actively and officially take part of the development effort (as opposed to FreeBSD [12] and OpenBSD [13] for instance). Besides the development model itself, the reasons identified here include the arguably rough, "old-school" aspect of the project releases for prospective users and developers.

A number of ideas and ways to address the issues listed are available, with various costs. Among them, extending releases to ready-to-flash software images for a range of devices and purposes is already an unofficial trend among the developers [14] (for the Raspberry Pi and BeagleBone for instance). An alternative installer with a graphical interface (as opposed to text-based) may also be welcome. Then, developers might expect more guidance as to their work environment, for which new possibilities are demonstrated. These goals led to the creation of the EdgeBSD Project [15], where there is complete freedom to experiment with these propositions.

Finally, functional devices with a modern, featureful user interface running on NetBSD in a variety of ways and purposes are introduced. They are based on the DeforaOS Project [16], which desktop environment can already be found packaged in the latest releases of the pkgsrc project [17]. They range from a typical desktop environment for users or developers, to embedded use as an e-book reader, a tablet [18], or as a telephony platform [19]. Software and hardware for the latter two projects was presented at BSD conferences in 2012 and 2013, and they are both being actively maintained [20], with significant progress to be demonstrated.

Development workflow

The development workflow of The NetBSD Project is handled separately when it comes to the base system and to package management. Although being subject to very different release-engineering policies and portability constraints, they are very closely related and a number of issues can be identified as a result.

Base system

Release-engineering

The NetBSD Project releases new major versions of the base system based on a given set of features desired for the release. A branch is forked from the main development tree once these features implemented, for heavier testing and preparation for the new release. This means that there is usually no particular deadline set with regard to when a new version should be released: it is done when considered ready.

There are three different kinds of releases [21]:

- major releases: like NetBSD 5.0 or 6.0, as described above;
- stable releases: like NetBSD 5.1 or 6.2, containing new features backported to the release branch;
- minor releases: like NetBSD 6.0.1 or 6.1.2, containing only essential fixes and security patches relevant to a stable release.

Stable branches for the two latest major releases are always maintained for essential fixes and security issues. In practice, this means that a stable branch for an old major release may be released after a new major release (for instance, NetBSD 5.2 was released after NetBSD 6.0). Although a common practice in the software industry, this can be confusing to some users.

More importantly, as of January 22nd 2014, there are at least five releases of the base system which are officially supported: 5.0.2, 5.1.3, 5.2.1, 6.0.3 and 6.1.2. While potentially useful for system administrators (who may then choose to follow minor release updates for stability) the number of releases available and supported at any given time can be overwhelming for the developers of the project and for users alike.

System sets

The NetBSD system is primarily shipped and distributed as a collection of binary sets. They simply consist of compressed tar archives. Two of them are essential for a working initial setup ("base" and "etc"), while the others provide additional functionality or documentation accordingly. Instructions and additional tools (where applicable) to assist the installation process are provided for each architecture supported.

Centralized development

The source code for NetBSD has always been maintained thanks to the CVS Source Code Management tool (SCM) [22]. CVS is a client-server based tool, which requires an active connection to the server to be able to commit to the repository. In other words it is a centralized system, requiring developers to be online to be able to track their own set of changes. Every developer is allowed to create and manage an own set of branches [23]. All branches are public and their use in CVS is resource-intensive, as it typically requires modifying the entire remote repository; as a result, their use is very limited.

More generally speaking, CVS is no longer on par with more modern SCM tools, and regularly criticized for its current limitations [24]. Moving or renaming files is a cumbersome process in CVS for instance. As another example, while in theory it is perfectly possible to use CVS in a decentralized fashion (using the "cvs import" command), in practice this is error-prone, resource intensive and inelegant, therefore seldom performed.

Package management

In NetBSD, packages are provided by the pkgsrc project [25]. pkgsrc is a cross-platform packaging system. Born in 1997 as a fork of the FreeBSD ports for NetBSD [26], it has largely diverged since then and supports over twenty platforms nowadays [27]. Although NetBSD remains the primary target platform for pkgsrc, the project is meant to be maintained separately and subject to distinct release-engineering policies.

Source-based approach

Contrary to NetBSD's own release management, pkgsrc is primarily meant as a source-based package deployment tool. A number of reasons behind this choice are regularly mentioned in the release notes [28]. Among them, emphasis is placed on the security aspects of this choice, which include the verification of the provenance and integrity of the source code, as well as that of the compilation environment. We do not believe these reasons to be accurate, given the possibility of the presence of backdoors in the original source code archives [29] or within the compilers used [30].

Quarterly releases

Again, the release management for pkgsrc is opposed to that of the NetBSD Project. pkgsrc is under constant development, with stable releases being tagged exactly four times per year (hence quarterly releases). After a short freeze period, used to work on the most essential build fixes prior to release, the main development tree is released as-is.

Centralized development

Just like for NetBSD, development of the pkgsrc project is performed with CVS as the SCM tool. It is therefore centralized as well, and hosted by The NetBSD Foundation on the same servers as NetBSD [31]. Importantly, being an official pkgsrc developer requires membership to The NetBSD Foundation too.

An additional repository for pkgsrc is available, called "pkgsrc-wip" for "Work In Progress". Managed by a prominent NetBSD and pkgsrc developer, Thomas Klausner (wiz@), this repository is hosted on SourceForge [32]. Also using CVS as the SCM tool, it is meant as a staging area for pkgsrc. Unlike its counterpart, it does not require membership to The NetBSD Foundation to be able to contribute. Packages from this repository are not included in official releases for pkgsrc though.

Issues identified

CVS is deprecated

The CVS SCM tool is no longer actively maintained, with the last preview version released about seven years ago. While mature, stable and functional, it is obsolete by today's standards and often dismissed by the new generation of software developers. We believe using CVS today to be a showstopper for the integration of new contributors to the NetBSD and pkgsrc projects for this reason.

Also, because of its centralized approach, it is very difficult for prospective developers to work efficiently without being official developers, and therefore gain the experience and confidence required to fulfill the integration process.

Migration to a newer SCM tool is a regular topic on the corresponding NetBSD mailing-list [33]. While no clear consensus has been made for a potential new tool, Git is regularly mentioned in such posts. The NetBSD wiki proposes a summary of these passionate discussions [34].

Lost contributions to the system

An indirect consequence of the centralized contribution management is the difficulty for external contributors to either submit significant contributions, or provide patches conveniently. In the first case in particular, such contributions have to be hosted on separate servers and are easily forgotten or worse, can be definitely lost [35].

Decentralized SCM tools like Git allow trivial ways to mirror external contributions locally, or handle patches conveniently.

Conflicting deployment and security policies between the base system and packages

The release policies of the base system and that of the packages are totally opposed on the following aspects:

- they are not synchronized (for either releases or end of support)
- the base system relies on binaries while packages are typically built from source.

As a result, it is difficult to run a stable system or to follow the latest packages. In the former case, packages are out of date after three to six months, and then are no longer supported for security. In the latter case, it is necessary to either build packages from source continuously (potentially re-building all of them regularly because of dependencies), or to switch releases of pkgsrc every three to six months, with many risks of regressions.

Both scenarios are commonplace, and the situation is worsened for every additional system (and architecture) the user is running. The significant amount of versions available for each major release of the base system also puts additional burden on the users.

We believe this situation to be counter-productive, harmful for the vitality of the project and dangerous for the security of its users.

Lack of quality assurance on pkgsrc releases

Packages in pkgsrc are maintained by individual developers, each potentially running a different system with a specific installed base. There is no way to ensure that any package will function as expected, even when installed on official, pristine binary releases. Pre-release periods for the pkgsrc project (called "freezes") are focused on build tests rather than functionality tests.

Another negative impact of the "freeze" periods is that it prevents developers to work on changes forbidden in these periods. This can also be seen as a consequence of the technical limitations of the centralized SCM tool in use, CVS (see the section called “Centralized development”).

Possibly insecure distribution of sets and packages

There are additional possibilities for external attacks, notably while distributing the repository of patches and checksums via insecure means. The most common way to distribute binary sets and packages is currently via HTTP, BitTorrent, anonymous FTP or rsync, with most protocols being unencrypted and easy to tamper with [36].

Source code is distributed in source sets, or optionally via separate means.

While SSH is supported, the CVS pserver protocol is still available and also vulnerable to such attacks.

Fortunately, there is a way to obtain a list of checksums applicable to the files officially available, that is also PGP-signed by NetBSD's Security Officer [37]. It is however neither straightforward to actually verify single downloads through this file, nor is this file actually signing the files being downloaded. Partial attacks against the hashing algorithms used might be enough to fool some users.

Binary signatures long broken

pkgsrc has been claiming to support signed binary packages since 2001. While being among the first software distributions to implement this feature, it was dysfunctional until recently: an issue with uninitialized variables let the integrity checks fail silently (and the package would then not install, with no error message). This issue was found and fixed while working on the EdgeBSD Project [38].

This illustrates the lack of interest of the project (and its current users) in the secure binary distribution of packages. Even though signed packages do not solve every issue in this regard, we believe they are a necessary step forward.

Relation to The NetBSD Project

Contributing to the pkgsrc project currently requires membership to The NetBSD Foundation. This has a number of consequences, including legal and political ones since The NetBSD Foundation is a legal entity in the United States [39].

Potential contributors may very well not desire (or not be allowed) to be part of The NetBSD Foundation, or to be associated with the NetBSD Project directly.

Conflicts with respect to other platforms

pkgsrc supports a significant number of platforms. These platforms do not all have the same features and API. This is generally not an issue: not every packages is required to build or work on each and every platform, and many issues can be patched and reported upstream.

However, there are cases where significant packages evolve at a faster pace than NetBSD does, and where they only support APIs which are not yet available in NetBSD. This situation leads to a dilemma:

- either packages are kept in their older versions, and still support NetBSD while being obsolete on other platforms,
- or on the contrary, the troublesome packages are updated but they no longer work on NetBSD, in spite of being the primary target platform.

This situation is unfortunately happening already: Xorg as packaged within pkgsrc (called "modular Xorg") was kept to an old version for some time, before being updated to a version requiring availability of the KMS API (Kernel Mode Switching). As a consequence, modular Xorg no longer works on NetBSD, and running Xorg on NetBSD requires obtaining X from the base system.

Package options are not binary-friendly

A number of pkgsrc packages support build-time options, of which default values depend on pkgsrc's default configuration, the current platform, and pkgsrc's global configuration file (`/etc/mk.conf`) if available. Packages depending on libraries built with different options are likely to be different and incompatible with each other. This can be troublesome when mixing official binary packages with packages built from sources, since the options chosen may as well differ. The list of options used for the binary packages may not even be available publicly. This is a commonplace scenario for users (like when building packages from wip) and we believe this harms reliable distribution of binary packages as well.

Moreover, some options may be essential to some users while being disabled by default (like LDAP support for instance). This also can easily discourage users from adopting pkgsrc (notably in corporate environments) since it makes it then necessary to maintain in-house binary package repositories. Again, we believe this to be harming the popularity of the pkgsrc project, especially in industrial environments.

This issue was solved differently in the Debian Project for instance. There, multiple versions of such packages are available, and all built automatically from the same source package. This design has been ruled out in pkgsrc because of its original source-based nature, but also because of the additional amount of work it would be expected to create while maintaining such packages [40][41].

Packages are heavy to download and install

Packages in pkgsrc are typically built directly from the source archive of a project, with only one package being built each time. These packages must therefore contain the development interface for any library they may provide for instance. This usually involves static libraries, development files, extensive documentation and additional files and binaries that may not be relevant in most cases. This means that packages are often bigger than they may just have to be.

While this issue can easily be dismissed on regular desktop and workstation systems (where the extra resources required may be negligible), it is not the case on embedded platforms. This, again, may discourage some industrial users to consider using pkgsrc for the software distribution of their packages.

Redistributable packages do not easily build unprivileged

While it is absolutely possible (and easy) to bootstrap pkgsrc to build packages for unprivileged users, this is not true of regular redistributable packages. They are currently expected to be built (or at least created) with root privileges, which almost always largely exceeds the privileges actually necessary for this task.

A workaround for this issue was found while working on the EdgeBSD Project, thanks to the **fakeroot** command from the sysutils/fakeroot package [42].

First list of suggestions

Switching to a decentralized SCM

As mentioned above in the section called “CVS is deprecated”, rather than CVS itself we believe the actual issue when it comes to the SCM tool to use is whether it allows external developers to use the repositories from a project. While a number of decentralized tools exist and provide their respective list of features and advantages, Git is by far the most popular tool today. Both GitHub and Gitorious are immensely popular, hosting millions of repositories [43]. Joyent, one of the largest industrial users of pkgsrc [44], also has its pkgsrc repositories hosted on GitHub.

A number of attempts have been made at providing functional repositories for The NetBSD Project using different SCM tools. Joerg Sonnenberger, a prominent NetBSD developer, is heavily involved in this task and evaluating alternatives to CVS [45]. His initial work was based on Git (in 2008) and then Fossil [46]. Since July 2011, he publishes mirrors for both the src and pkgsrc source trees on the GitHub platform.

These mirrors have a significant number of users, with the pkgsrc repository forked about 50 times on GitHub alone [47].

Provide a public SCM service for any potential contributor

It should be possible to avoid losing contributions (as mentioned above in the section called “Lost contributions to the system”) by providing a mirror of the source code repositories, where just about anyone could publish code as well.

This may cause indirect legal or security issues, as there would be a possibility for contributors to upload tainted code, malicious files or otherwise inappropriate content (which caused a premature end to no less than the Google Code Download Service [48]).

However, the popularity of this platform should also attract moderators in sufficient proportions. Additionally, some SCM tools allow for the efficient access control and removal of branches (like Git when combined with Gitolite [49]).

In any case, this service (or the external branches) should be handled separately from the main repositories, or clearly advertized as such.

Long-Term-Support (LTS) branches for pkgsrc

It appears necessary to maintain a branch of pkgsrc for both stability and security. A proposal has already been made (and declined) to the PMC to keep maintaining the release corresponding to the first quarter of every (second) year in this fashion.

This process is commonplace in major Open Source projects and software distributions (like Mozilla ESR and Ubuntu LTS) but does involve an amount of extra work, on which ground it was not adopted. One such additional task is that of maintaining a separate security vulnerability list for each LTS branch.

Rolling-release for the stable branches

It would likely help both reduce the maintenance work, please every type of user and help acceptance of minor issues to maintain only two stable releases per major release. They would consist of:

- a branch remaining as close as possible to the original major release as possible (ie accepting security and essential fixes only)
- a branch with every security issue, fix or backported addition deemed fit by the developer requesting the pull-up and the release-engineering team, in a rolling-release fashion (e.g. with updated binaries always available) and an official release-worthy tag and binaries for the latest essential change pushed.

The first choice is mandatory, because it is the safest way to allow building packages compatible with every later addition to the release. It is therefore meant for bulk builders, and for administrators requiring absolute stability to the system while also tracking security fixes.

The second choice would also have the advantage to more easily provide an update channel for the major release, and probably encourage users to use and provide feedback about the same, updated version. NetBSD already provides such a channel [50], but it is not officially available and usually not mirrored on other servers. There is no official in-place upgrade tool either, although a few are available already (like sysupgrade in pkgsrc [51]).

Towards industry standards: EdgeBSD

EdgeBSD is a young project started in the second half of 2013 [52] by Pierre Pronchery, already a NetBSD developer at the time. It was started as a way to experiment freely with changing some of the usual aspects of working with The NetBSD Project, and hopefully attracting new developers to its ecosystem.

From portability to usability

The main strength of NetBSD is certainly how much its developers care for a clean and intelligible design. This has eventually allowed the system to be easily portable, and gained the project a reputation for portability. Among these capacities, we would like to emphasize on the following:

- the system is cross-compiled by default,
- internal frameworks are carefully integrated and documented,
- hardware-level bus access is abstracted away for driver developers.

Unfortunately, in spite of these unique capabilities, improving the system for use on desktop environments has never gained much attraction (or, rather, acceptation [53]).

One of the main reasons behind this situation is certainly the amount of work required to track the latest developments of the broader Open Source desktop community, as typically driven by the GNU/Linux class of systems. Most of the recent developments require major architectural updates to the system in order to work (or even compile) optimally, like KMS (Kernel Mode Switching) with Xorg. Desktop environments take time to fully port as well (GNOME 3, XFCE), while some changes may not be desirable in the first place (like systemd, another init system).

It was demonstrated that it is indeed possible to provide a modern and stable desktop environment on top of NetBSD, on desktop and embedded environments regardless (including tablets [54] or possibly smartphones [55]). Projects like the DeforaOS desktop [56] aim at running and integrating on more systems than just GNU/Linux.

In fact, EdgeBSD may go as far as adopting a default desktop environment, providing both a controlled and maintained user experience and a reference implementation for other projects to work with. Another very important reason for this is the availability of an Integrated Development Environment within this desktop, providing developers with a known, stable and featureful work environment that is more easily supported.

One way to help this happen is to provide prospective users and developers with ready-to-use system images for a number of devices and contexts; this is also a goal of the EdgeBSD Project.

Decentralized development workflow

Given the increasing popularity of Git within the industry (as can be seen in the Android ecosystem) it was decided to use this SCM tool to host its repositories. As of today, the initial fork was obtained from Joerg Sonnenberger's work directly on GitHub. A recurring issue was found with this approach: the Git chain of commits is regularly rewritten.

Incremental updates to the Git repository format sometimes break the existing chain of commits and are published using "forced pushes". In Git, it is typically impossible to go back in time. This is however common practice among NetBSD developers through the use of the "cvs admin" command, usually to improve commit messages after the fact. In turn, this may force Git users to perform tedious manual operation.

This issue is currently being addressed: given the popularity of the Git mirrors, NetBSD's PMC - pkgsrc Management Committee - has disallowed the use of the "cvs admin" command in the pkgsrc repository. This should tremendously ease conversions from now on, although issues with the "cvs import" command may still occur.

---

[1] Initial revision for `src/Makefile`, *http://cvsweb.netbsd.org/bsdweb.cgi /src/Makefile?rev=1.1&content-type=text/x-cvsweb-markup*

[2] The NetBSD Project, *http://www.netbsd.org/*

[3] « What is NetBSD? » in the NetBSD Guide, *https://www.netbsd.org/docs/guide/en/chap-intro.html*

[4] The EdgeBSD Project, *https://www.edgebsd.org/*

[5] The DeforaOS Project, *https://www.defora.org/*

[6] Open Sources: Voices from the Open Source Revolution, *http://oreilly.com/catalog /opensources/book/kirkmck.html*

[7] The NetBSD system, *https://www.netbsd.org/about/system.html*

[8] Linux distribution, *https://en.wikipedia.org/wiki/Linux_ distribution*

[9] « NetBSD development model » in EdgeBSD at FrOSCon 2013, *http://people.defora.org /~khorben/papers/froscon2013/EdgeBSD.pdf*

[10] « Criticism » in Concurrent Versions System, *https://en.wikipedia.org /wiki/Concurrent_ Versions_ System#Criticism*

[11] Products based on NetBSD, *http://www.netbsd.org/gallery/products.html*

[12] « Who Uses FreeBSD? » in the FreeBSD Handbook, *http://www.freebsd.org /doc/en_ US.ISO8859-1/books/handbook/nutshell.html#introduction-nutshell-users*

[13] « Commercial Users » in OpenBSD Users, *http://www.openbsd.org/users.html*

[14] Jared D. McNeill's misc repository on NetBSD's FTP server, *http://ftp.netbsd.org /pub/NetBSD/misc/jmcneill/rpi/*

[15] The EdgeBSD Project, *https://www.edgebsd.org/*

[16] The DeforaOS Project, *https://www.defora.org/*

[17] The deforaos-desktop meta-package for pkgsrc, *http://cvsweb.netbsd.org/bsdweb.cgi/pkgsrc /meta-pkgs/deforaos-desktop/*

[18] Touch your NetBSD at EHSM 2012, *http://people.defora.org/~khorben/papers/ehsm2012/Touch%20your%20NetBSD.pdf*

[19] Call your NetBSD at BSDCan 2013, *http://www.bsdcan.org/2013/schedule/events/381.en.html*

[20] The DeforaOS Open Source Project on Ohloh, *http://www.ohloh.net/p/DeforaOS*

[21] NetBSD release glossary and graphs, *http://www.netbsd.org/releases/release-map.html*

[22] Concurrent Versions System on Wikipedia, *http://en.wikipedia.org/wiki/Concurrent_ Versions_ System*

[23] src/doc/BRANCHES, *http://cvsweb.netbsd.org/bsdweb.cgi/~checkout~/src/doc/BRANCHES?content-type=text/plain*

[24] Concurrent Versions System on Wikipedia: Criticism, *http://en.wikipedia.org/wiki/Concurrent_ Versions_ System#Criticism*

[25] pkgsrc, *http://www.pkgsrc.org/*

[26] 10 years of pkgsrc - pkgsrc and the concepts of package management 1997-2007, *https://www.netbsd.org/gallery/10years.html*

[27] pkgsrc: The NetBSD Packages Collection, Supported platforms, *http://www.netbsd.org/docs/software/packages.html#platforms*

[28] pkgsrc-2013Q4 branched, *http://mail-index.netbsd.org/pkgsrc-users/2013/12/31/msg019107.html*

[29] BitchX Trojan Horse Vulnerability, *https://www.juniper.net/security/auto/vulnerabilities/vuln7333.html*

[30] Strange Loops: Ken Thompson and the Self-referencing C Compiler, *http://scienceblogs.com/goodmath/2007/04/15/strange-loops-dennis-ritchie-a/*

[31] NetBSD CVS Repositories, *http://cvsweb.netbsd.org/bsdweb.cgi/*

[32] The pkgsrc-wip project, *http://pkgsrc-wip.sourceforge.net/*

[33] The first step away from CVS on tech-repository, *http://mail-index.netbsd.org/tech-repository/2010/01/06/msg000204.html*

[34] tech-repository on the NetBSD Wiki, *http://wiki.netbsd.org/mailing-lists/tech-repository/*

[35] Rubberhose mirror (at the Internet Archive), *http://web.archive.org/web/20110726185300/http://iq.org/~proff/rubberhose.org/*

[36] NetBSD Mirror Sites, *http://www.netbsd.org/mirrors/*

[37] NetBSD-6.1.3_hashes.asc, *ftp://ftp.netbsd.org/pub/NetBSD/security/hashes/NetBSD-6.1.3_ hashes.asc*

[38] NetBSD Problem Report #48194, *http://gnats.netbsd.org/48194*

[39] The NetBSD Foundation, Inc., *http://www.netbsd.org/foundation/*

[40] "New options for freeswitch" thread on the "tech-pkg" mailing-list, *http://mail-index.netbsd.org/tech-pkg/2012/10/17/msg010217.html*

[41] "Package split or package options?" thread on the "tech-pkg" mailing-list, *http://mail-index.netbsd.org/tech-pkg/2013/12/04/msg012303.html*

[42] "pkgsrc/mksandbox.sh" from the EdgeBSD Project, *http://git.edgebsd.org/gitweb /?p=edgebsd-infrastructure.git;a=blob;f=pkgsrc/mksandbox.sh*

[43] 10 million repositories, GitHub, *https://github.com/blog/1724-10-million-repositories*

[44] Joyent Packages Documentation, *http://pkgsrc.joyent.com/*

[45] Fossil and NetBSD, *http://www.sonnenberger.org/2010/10/24/fossil-and-netbsd/*

[46] Fossil, *http://www.fossil-scm.org/*

[47] The pkgsrc network graph on GitHub, *https://github.com/jsonn/pkgsrc/network*

[48] A Change to Google Code Download Service, *http://google-opensource.blogspot.de/2013/05 /a-change-to-google-code-download-service.html*

[49] Gitolite: Hosting git repositories, *http://gitolite.com/gitolite/*

[50] Daily builds of the netbsd-6 branch on `ftp.netbsd.org`, *http://nyftp.netbsd.org /pub/NetBSD-daily/netbsd-6/*

[51] Introducing sysupgrade for NetBSD, *http://julipedia.meroh.net/2012/08/introducing-sysupgrade.html*

[52] "EdgeBSD was introduced at FrOSCon" 2013, *https://www.edgebsd.org/edgebsd/news/6 /edgebsd%20was%20introduced%20at%20froscon*

[53] Archives for the netbsd-desktop mailing-list, *http://mail-index.netbsd.org/netbsd-desktop/*

[54] "Touch your NetBSD" presentation at EHSM 2012, *http://mail-index.netbsd.org/netbsd-advocacy/2013/01/13/msg000512.html*

[55] "Call your NetBSD" presentation at BSDCan 2013, *http://www.bsdcan.org/2013/schedule /events/381.en.html*

[56] "Graphical environment" from DeforaOS, *http://www.defora.org/os/wiki /3426/graphical%20environment*