EuroBSDCon 2004

NetBSD/Desktop: Scalable Workstation Solutions

Jan Schaumann

`jschauma@{cs.stevens.edu,netbsd.org}`

```
 jschauma@netbsd.org: 136D 027F DC29 8402 7B42  47D6 7C5B 64AF AF22 6A4C
jschauma@cs.stevens.edu: 4C58 6D3F B5F7 5E1E 4B5A  EC21 1F7D B5CE B80E 83A5
```

Department of Computer Science

STEVENS Institute of Technology

Introduction

Let’s debunk the myth that NetBSD is “not ready for the desktop”!

We will:

- look at general requirements of a Desktop System

Introduction

Let’s debunk the myth that NetBSD is “not ready for the desktop”!

We will:

- look at general requirements of a Desktop System
- consider specific requirements of the environment in question

Introduction

Let’s debunk the myth that NetBSD is “not ready for the desktop”!

We will:

- look at general requirements of a Desktop System
- consider specific requirements of the environment in question
- analyze what constitutes “user-friendliness”

Introduction

Let’s debunk the myth that NetBSD is “not ready for the desktop”!

We will:

- look at general requirements of a Desktop System
- consider specific requirements of the environment in question
- analyze what constitutes “user-friendliness”

- think about how to maintain large numbers of identical desktop systems

Introduction

Let’s debunk the myth that NetBSD is “not ready for the desktop”!

We will:

- look at general requirements of a Desktop System
- consider specific requirements of the environment in question
- analyze what constitutes “user-friendliness”

- think about how to maintain large numbers of identical desktop systems
- present a scalable framework in production use

Introduction

Let’s debunk the myth that NetBSD is “not ready for the desktop”!

We will:

- look at general requirements of a Desktop System
- consider specific requirements of the environment in question
- analyze what constitutes “user-friendliness”

- think about how to maintain large numbers of identical desktop systems
- present a scalable framework in production use
- analyze strengths and weaknesses

Introduction

Let’s debunk the myth that NetBSD is “not ready for the desktop”!

We will:

- look at general requirements of a Desktop System
- consider specific requirements of the environment in question
- analyze what constitutes “user-friendliness”

- think about how to maintain large numbers of identical desktop systems
- present a scalable framework in production use
- analyze strengths and weaknesses
- conclude wisely (and timely)

Requirements of a user-friendly Desktop System

- in general:
  - common hardware must be supported
  - primary use as a *desktop* $\longrightarrow$ certain “standard” applications must be available
  - *user*-friendly $\longrightarrow$ knowledge and consideration of the users needs

Requirements of a user-friendly Desktop System

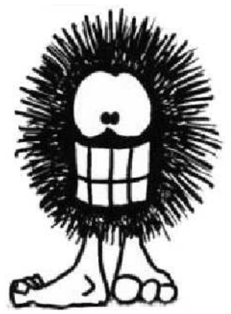
- in general:
  - common hardware must be supported
  - primary use as a *desktop* $\longrightarrow$ certain “standard” applications must be available
  - *user*-friendly $\longrightarrow$ knowledge and consideration of the users needs

- specificially, in this environment:
  - academic target userbase
  - users vary widely in experience, knowledge, interest
  - large number of different applications needs to be available
  - several key applications are required
  - open source a plus

Requirement: User-friendly

Common understanding of user-friendliness:

- easy OS installation
- easy software installation / upgrades
- safe and reliable handling of security patches
- installed software is easy to use

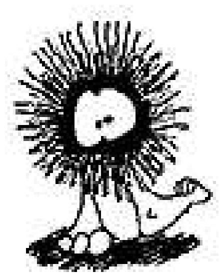


Requirement: User-friendly

Common understanding of user-friendliness:

- easy OS installation
- easy software installation / upgrades
- safe and reliable handling of security patches
- installed software is easy to use

But: most of these tasks are not – and *should* not – be performed by the *user*!



Requirement: User-friendly

Common **mis**understanding of user-friendliness:

- ~~easy OS installation~~
- ~~easy software installation / upgrades~~
- ~~safe and reliable handling of security patches~~
- installed software is easy to use

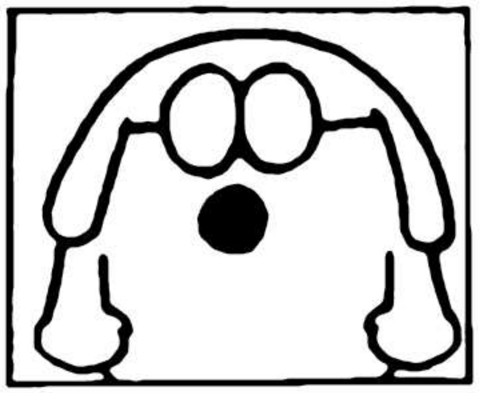
In other words, user-friendliness means:

- applications the user needs are installed
- consistent environment across networked machines
- users need *not* know or notice which particular OS is chosen
- hide unneccessary complexities
- don’t restrict advanced users

Requirement: Admin-friendly

An **admin**-friendly OS allows for:

- easy OS installation
- easy software installation / upgrades
- safe and reliable handling of security patches



Requirement: Admin-friendly

An **admin**-friendly OS allows for:

- easy OS installation / upgrades
  - hardware support
  - capable of non-interactive and/or customized installation
  - complete source tree
  - reliable release engineering
- easy software installation / upgrades
  - mature package management system
  - support for (proprietary) third-party vendor applications
- safe and reliable handling of security patches

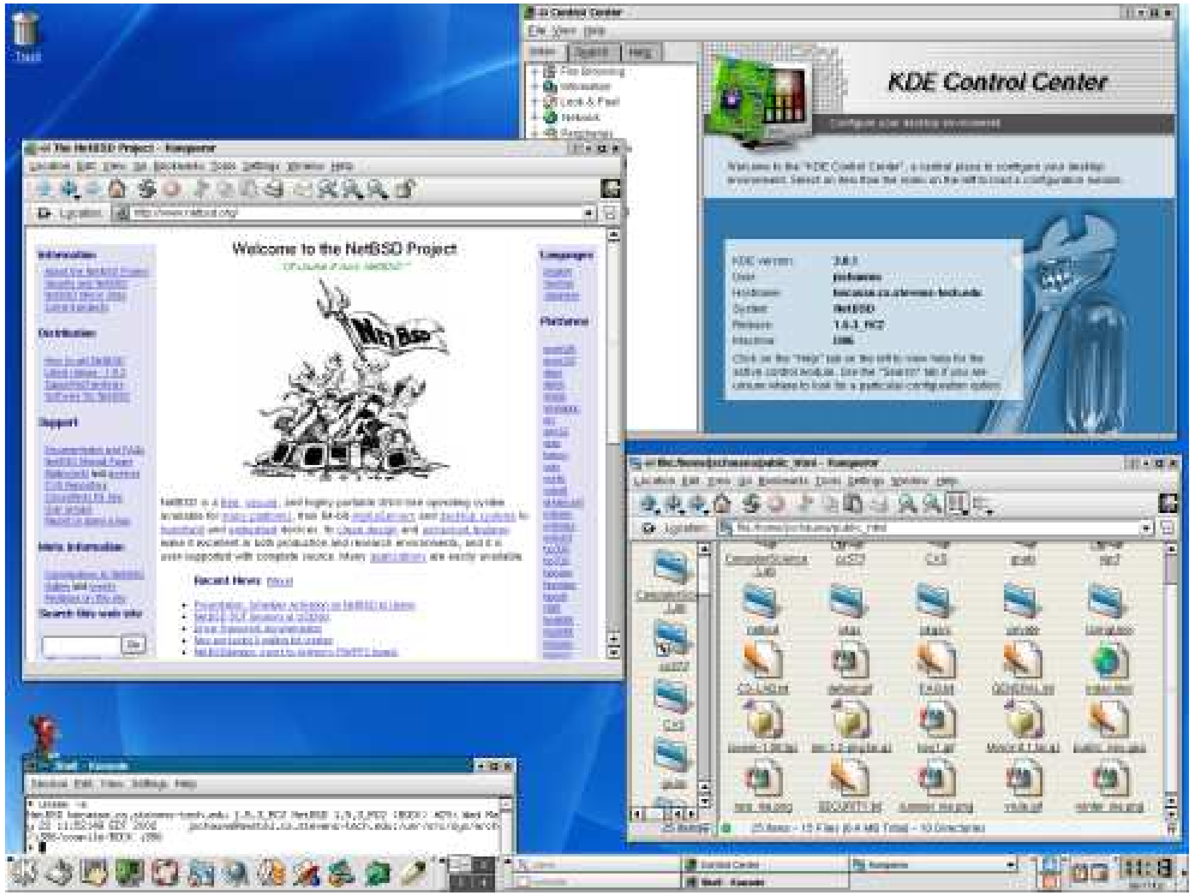
Choosing a Desktop System

Consider:

- general user requirements
- specific requirements posed by the target user base
- number of applications installed
- type of applications installed
- number of desktop systems deployed
- potential expansion

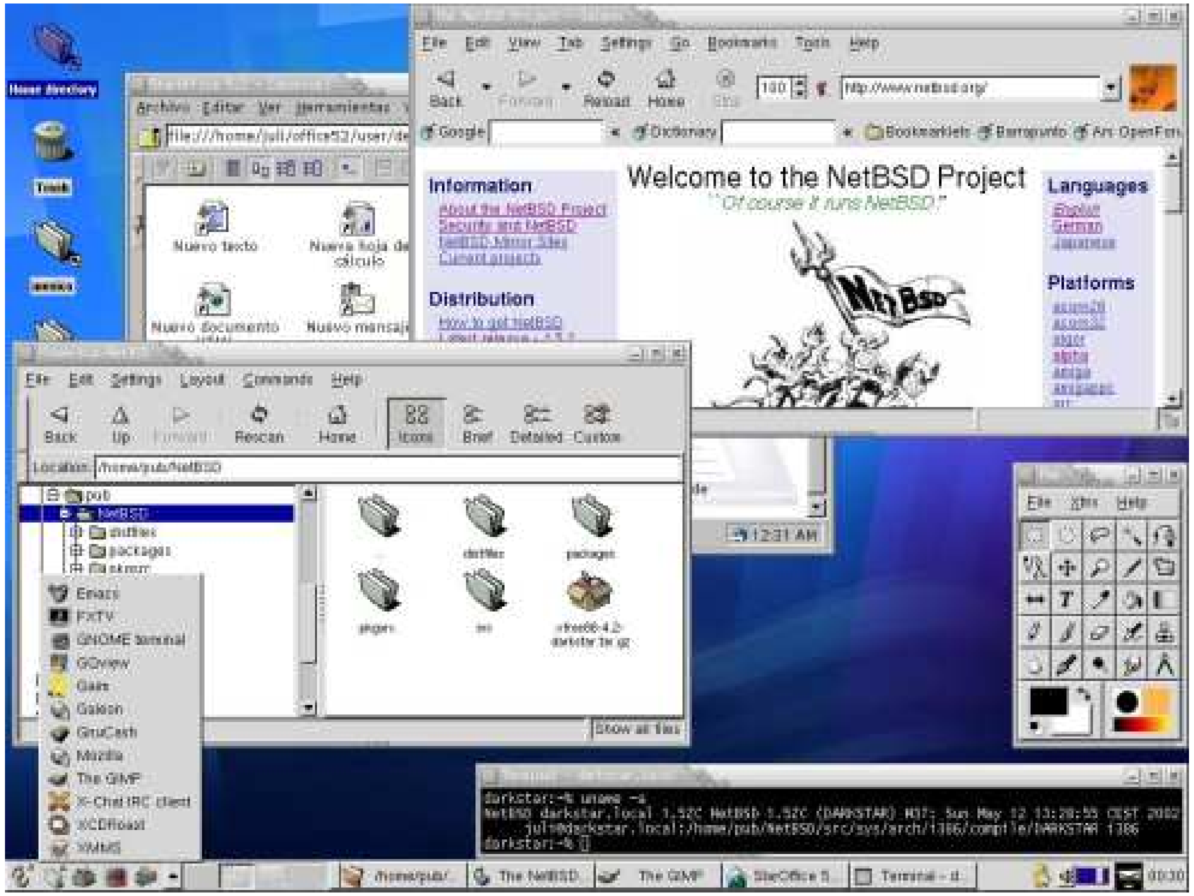
So... why NetBSD?

It’s friendly to users who like KDE:



So... why NetBSD?

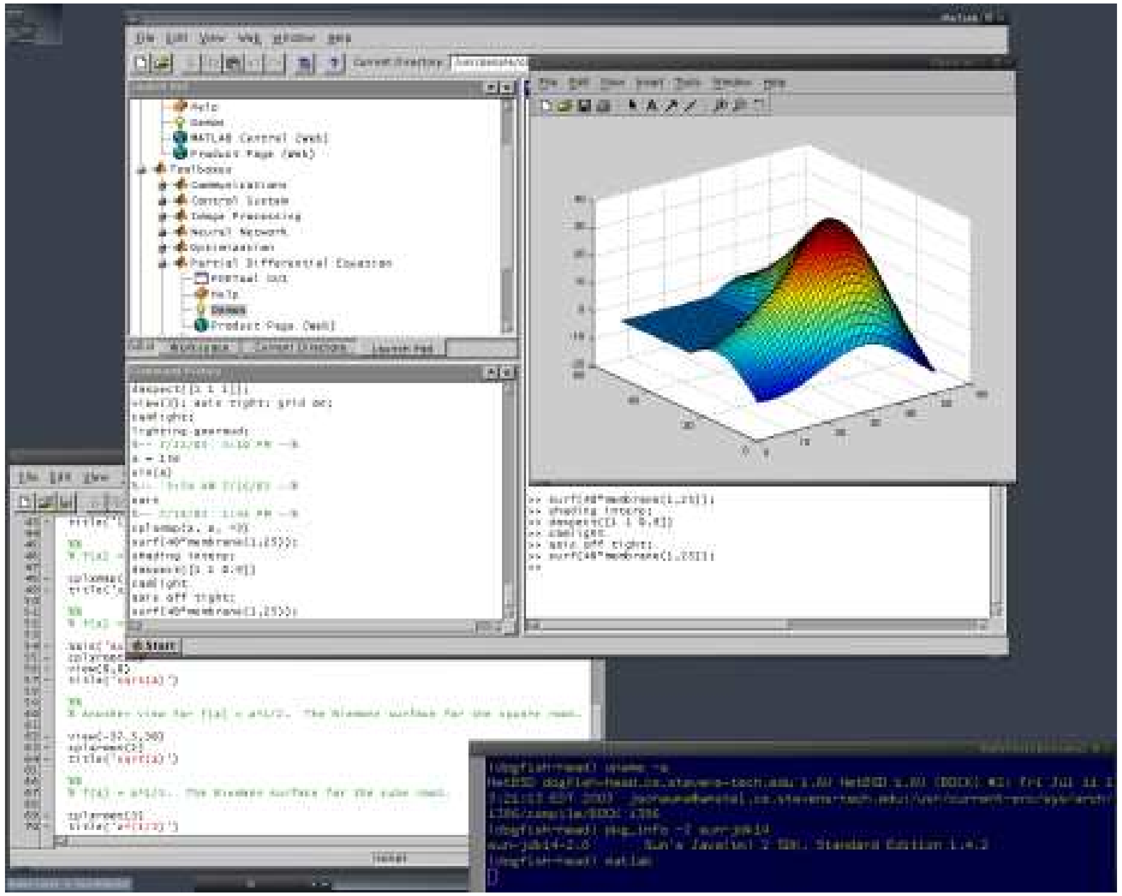
It’s friendly to users who like GNOME:



```
darkstar:~% uname -a
NetBSD darkstar.local 1.52C NetBSD 1.52C (DARKSTAR) #37: Sun May 12 13:28:55 CEST 2002
     jul@darkstar.local:/home/pub/NetBSD/src/sys/arch/i386/compile/DARKSTAR i386
darkstar:~%
```

So... why NetBSD?

It's friendly to users who need specific applications:

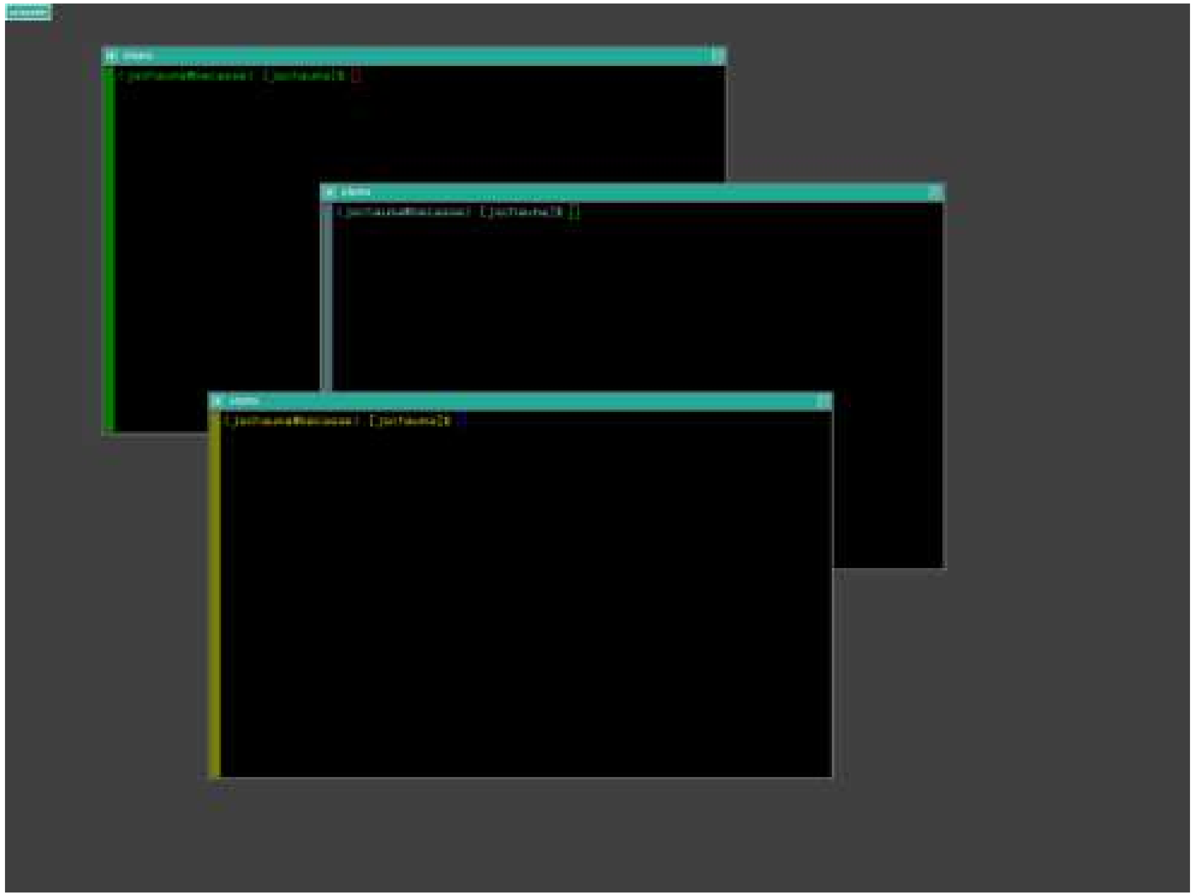


```
>> surf(40*membrane(1,25));
>> shading interp;
>> daspect([1 1 0.8])
>> camlight
>> axis off tight;
>> surf(40*membrane(1,25));
>>
```

```
[dogfish-head]: uname -a
NetBSD dogfish-head.cs.stevens-tech.edu 1.6U NetBSD 1.6U (DOCK) #1: Fri Jul 11 1
7:21:13 EDT 2003  jschauma@[illegible].cs.stevens-tech.edu:/usr/current-src/sys/arch/
i386/compile/DOCK i386
[dogfish-head]: pkg_info -I sun-jdk14
sun-jdk14-2.0         Sun's Java(tm) 2 SDK, Standard Edition 1.4.2
[dogfish-head]: matlab
```

So... why NetBSD?

It’s friendly to users who prefer a clean desktop:



So... why NetBSD?

Alright, we get it, **NetBSD is user-friendly**! What else?

So... why NetBSD?

Alright, we get it, **NetBSD is user-friendly**! What else?

Why, **NetBSD is admin-friendly**, too. Of course!

- complete OS (kernel and userland from one source tree)
- stable releases known to actually *be* stable
- releases are supported for a long time
- new versions only released when they're really ready
- thousands of third-party applications available
- incredible cross-platform package management using *pkgsrc*
- support for required proprietary applications
- great reference platform for Computer Science students

Case Study: NetBSD Desktops at Stevens Institute of Technology

Infrastructure:

- large number of NetBSD workstations
  - public laboratories
  - faculty desktops
  - research facilities
- (almost) identical hardware
  - easy replacement of faulty hardware
- identical software installation
  - users can move from machine to machine
  - replacing hardware does not mean users can't access their work environment

NetBSD/Desktop @ Stevens: Infrastructure (cont'd)

- central build server
  - easy maintenance of workstation image
  - one place to build and install software
  - server initiated push-strategy updates clients
- In numbers:

# of administrative scripts	7
total LOC of administrative scripts	388
# of users	approx. 2900
# of workstations	70
# of third-party packages not under pkgsrc	7
# of third-party packages under pkgsrc	1054
size of workstation image	9.3 GB

NetBSD/Desktop @ Stevens: Infrastructure (cont'd)

Server Setup:

- plenty of RAM and disk space
- full source tree
- three pkgsrc trees:
  - latest stable (ie *pkgsrc-2004Q3*)
  - *HEAD*
  - custom
- two chroots:
  - `/sandbox`
  - `/new`
- `audit-packages(8)` run nightly
- allows `syslog`ging of workstations over IPSec
- talks to clients via `rsh(1)`+IPSec or `ssh(1)`

NetBSD/Desktop @ Stevens: Infrastructure (cont'd)

Client Configuration:

- `/new` (on server) contains full image, including over 1000 third-party applications
- very few files need to be unique to each host:

```
etc/X11/XF86Config
etc/master.passwd
etc/racoon/psk.txt
etc/rc.conf
etc/spwd.db
etc/ssh/ssh_host*_key.admin{,.pub}
etc/printcap
```

NetBSD/Desktop @ Stevens: Software Installation

In general:

- all software installed if at all possible via the NetBSD Packages Collection
- new packages created as necessary
- non-pkgsrc'd software goes into `/usr/local`

In detail:

- if software is not available in latest stable branch
  - software available in *HEAD*:
    - merge by hand into custom tree
  - software **not** available in *HEAD*:
    - create package in *HEAD*
    - commit package to pkgsrc
    - add new package into custom tree

NetBSD/Desktop @ Stevens: Software Installation (cont'd)

Installation procedure:

```
$ chroot /sandbox
# make install package
# exit
$ chroot /new
# pkg_add <newpackage>
```

Upgrade procedure:

```
$ chroot /sandbox
# make update package
# exit
$ chroot /new
# pkg_add -u <newpackage>
```

Important variables in `/etc/mk.conf`:

```
IGNORE_RECOMMENDED=YES
DEPENDS_TARGET=package
```

NetBSD/Desktop @ Stevens: Update Procedure

The script `push.sh` updates a given workstation:

- push initiated by server
- `rsync(1)` via either IPSec'd `rsh` or SSH on a separate port using a secret key
- sync in individual passes
- allow for site-wide exclusions
- allow for per-host exclusions and "absolute" files
- execute initial or final commands on each remote host

NetBSD/Desktop @ Stevens: Update Procedure (cont'd)

Security considerations:

- push initiated by server
  - no client can "request" an update
- `rsync(1)` via either IPSec'd `rsh` or SSH on a separate port using a secret key
  - clients are authenticated
  - sensitive files are transferred encrypted

NetBSD/Desktop @ Stevens: Update Procedure (cont'd)

Times for a single push:

type	remote shell	time
minor update	rsh + ipsec	5.5m
minor update	ssh	7.6m
major update	rsh + ipsec	7.9m
major update	ssh	8.25m
full update	rsh + ipsec	21.8m
full update	ssh	23.8m

- *minor update*: update of a few files or a small package
- *major update*: update of at least three large packages (such as mozilla, KDE etc.)
- *full update*: update of the entire userland and several large packages

NetBSD/Desktop @ Stevens: Update Procedure (cont'd)

Times for an update of all workstations:

type	remote shell	time
minor update	rsh + ipsec	36m
minor update	ssh	49m
major update	rsh + ipsec	47m
major update	ssh	51m
full update	rsh + ipsec	149m
full update	ssh	155m

- *minor update*: update of a few files or a small package
- *major update*: update of at least three large packages (such as mozilla, KDE etc.)
- *full update*: update of the entire userland and several large packages

NetBSD/Desktop @ Stevens: New Workstation Installation

The basic steps to integrate a new workstation into this setup are trivial:

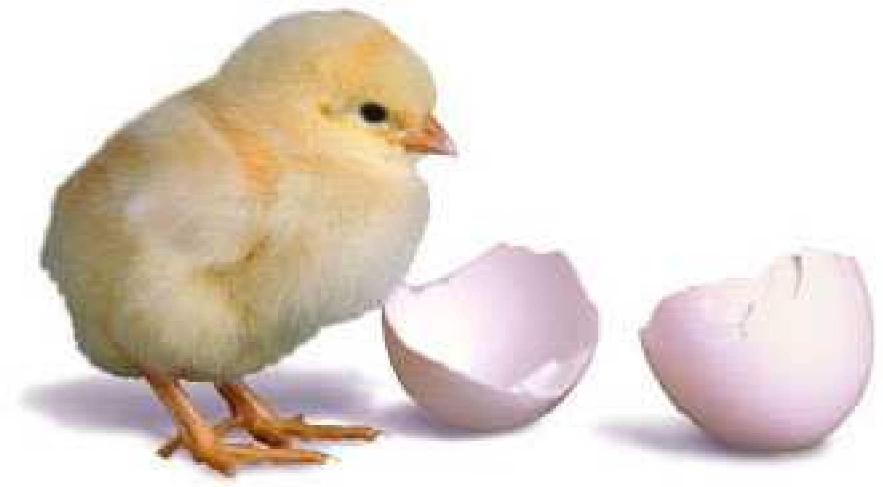
- create the necessary host-specific configuration files on the server (hostname, unique IKE PSK, ssh keys, X11 configuration, printcap etc.)
- install new host
    - boot off standard install media
    - `disklabel(8)` and `newfs(8)` disk
    - configure network
    - nfs mount client image from server
    - `pax(1)` over data
    - install bootblocks

NetBSD/Desktop @ Stevens: New Workstation Installation (cont'd)

Security considerations:

1. certain files must not be transmitted in the clear, so must not be in the client image and can not be installed initially
2. some of these exact files are necessary to allow subsequent pushes

Hmmm...



NetBSD/Desktop @ Stevens: What’s next?

Improving the general setup:

- consider switching to *pkgviews* framework
- consider bulk-building the set of installed packages on a regular basis from both stable branch and HEAD
- consider running bulk-builds of *all* packages to ensure availability of binary packages
- consider creation of custom install CDs (possibly per host)
- place client-specific configuration files under CVS control
- consider CVS or similar approaches for entire workstation image
- provide documentation... oh, wait... done! :-)

NetBSD/Desktop @ Stevens: What’s next?

Improve Security:

- consider mapping IPs and MAC addresses
- consider asymmetric cryptography approach to chicken-and-the-egg problem
- consider use of install-key to encrypt the few sensitive files for installation process
- perform installations on private network only

Conclusion

NetBSD is

- “ready for the desktop”
- *user*-friendly
- *admin*-friendly

The framework presented:

- is simple yet flexible
- can easily be deployed
- has (of course) room for improvement

Questions? Comments?

Thanks!