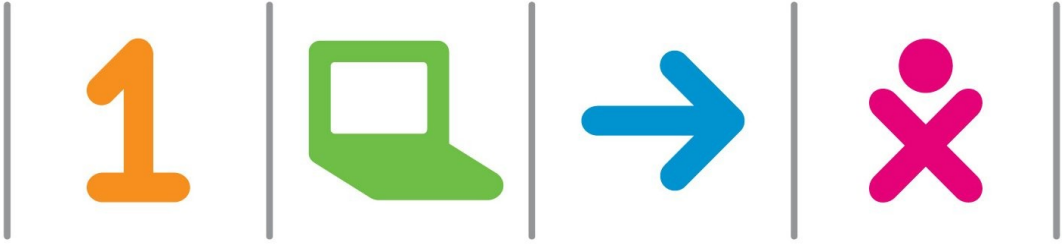


One Laptop per Child

Bringing OLPC to the GNU/Linux Desktop

Dr. C. Scott Ananian <cscott@laptop.org>

Talk Outline

- OLPC Overview
  - What we want to do
  - How we're doing
- Areas of divergence
  - Networking
  - Sugar & Activities
  - Security
  - Power Management
  - **Data store**
- Let's work together!

Sometimes the riskiest path is the status quo.

A global transformation of education

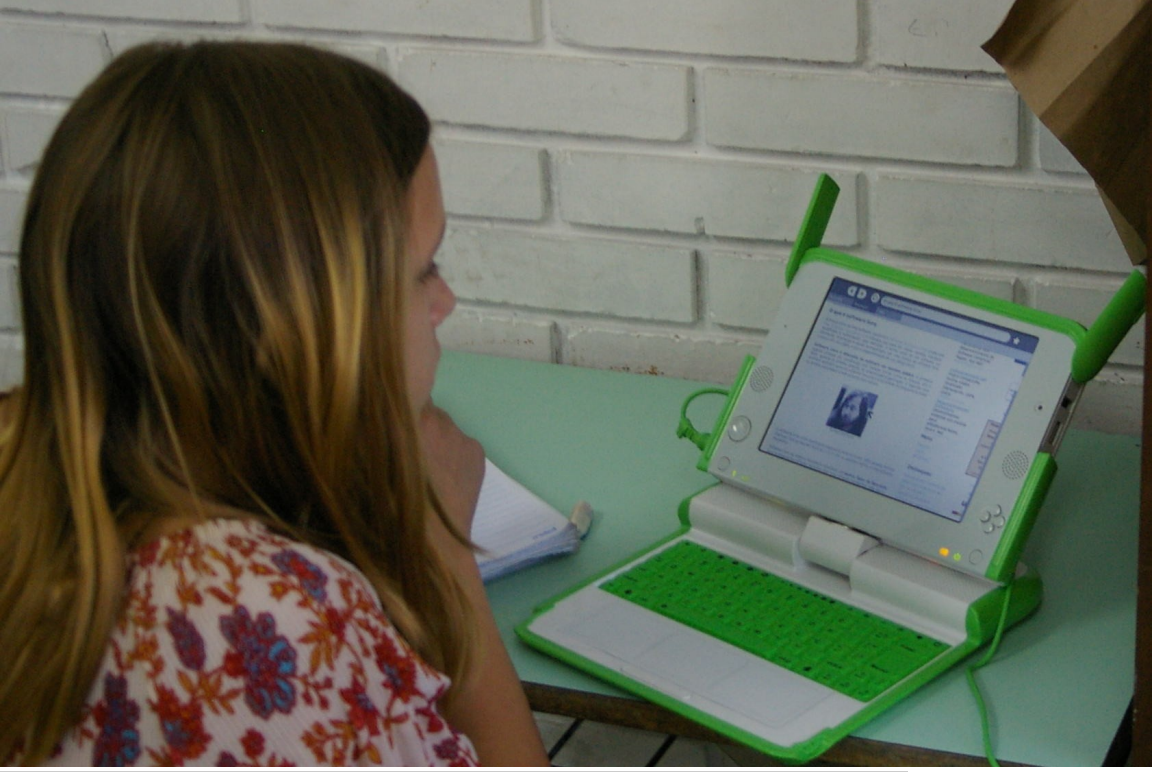
- It's about giving children who don't have the opportunity for learning that opportunity: so it's about access; it's about equity; and it's about giving the next generation of children in the developing world a bright and open future.

Children lack opportunity, not capability

1. High-quality education for every child is essential to provide an equitable and viable society;
2. A connected laptop computer is the most powerful tool for knowledge creation;
3. Access on a sufficient scale provides real benefits for learning.

ONE LAPTOP PER CHILD

ONE LAPTOP PER CHILD



ONE LAPTOP PER CHILD

A vaccine is an agent of change.

Jonas Salk made the analogy between education reform and immunology: both require scale and reach in order to be successful.

A connected laptop is not a cure

- ...but it is an agency through which children, their teachers, their families, and their communities can manufacture a cure.
- They are tools with which to think, sufficiently inexpensive to be used for work and play, drawing, writing, and mathematics.

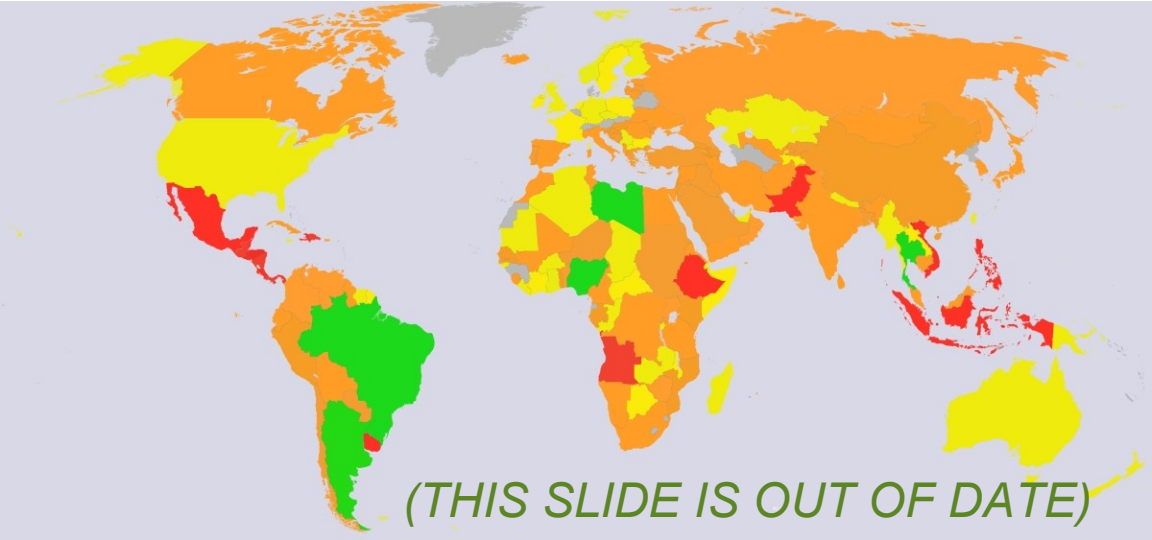
Three traits we humans all share

1. we learn (and teach);
2. we express; and
3. we are social.

Five principles

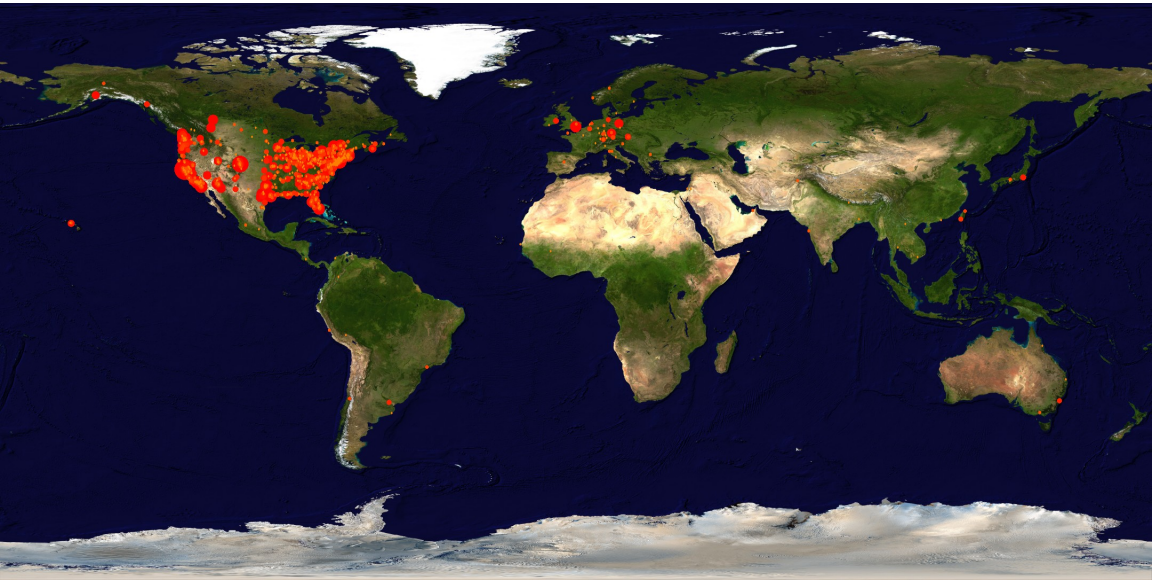
1. child ownership—use of the laptop at home;
2. low ages—ages 6 to 12—low floor, no ceiling;
3. saturation and
4. connection—collaborative and community;
5. free and open—the child is an active participant in a global learning community.

Where?



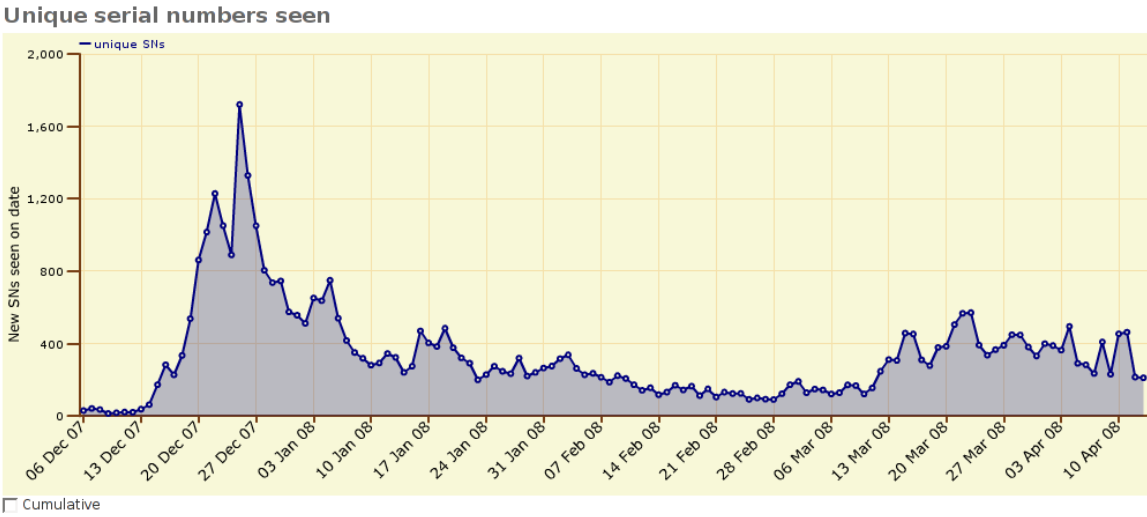
(green) launch; (red) 2nd wave; (orange) ministerial interest; (yellow) NGO interest

G1G1 geolocation

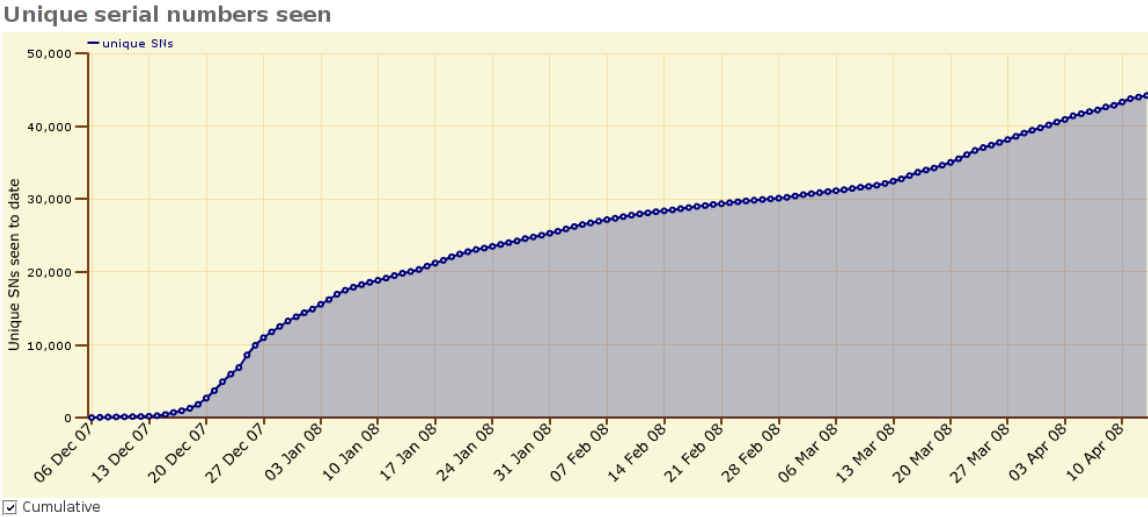


Caveat: This data is somewhat bogus.

Unique SNs per day



Unique SNs (cumulative)



Talk Outline

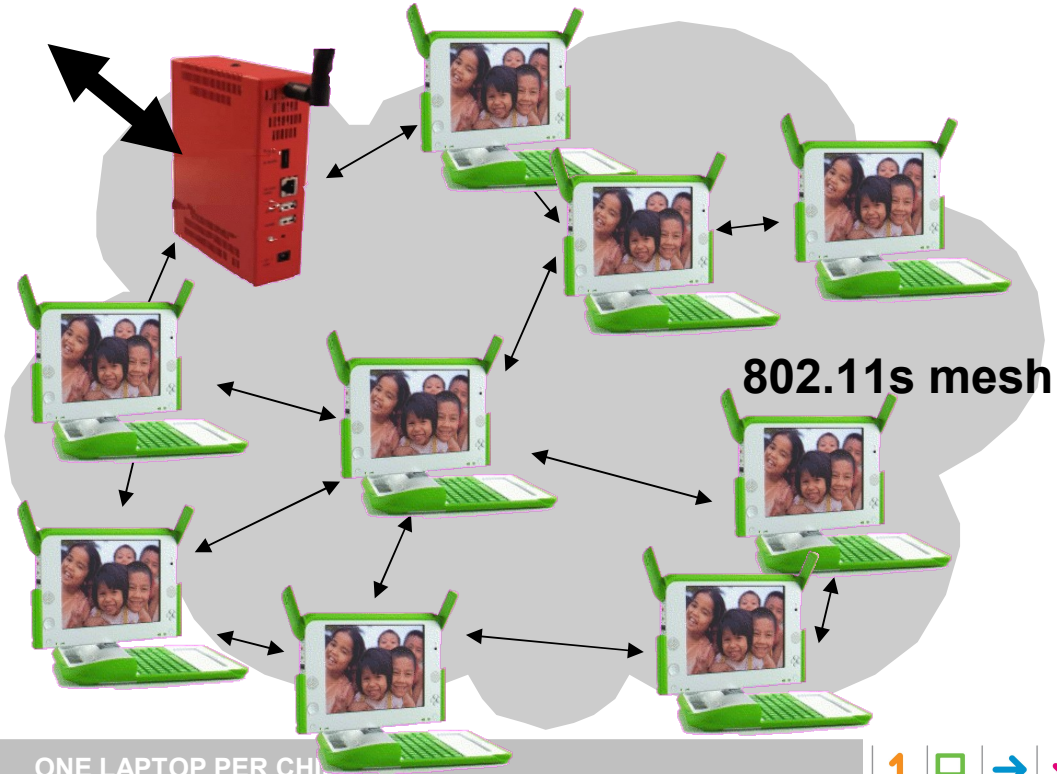
- OLPC Overview
  - What we want to do
  - How we're doing
- Areas of divergence
  - Networking
  - Sugar & Activities
  - Security
  - Power Management
  - **Data store**
- Let's work together!

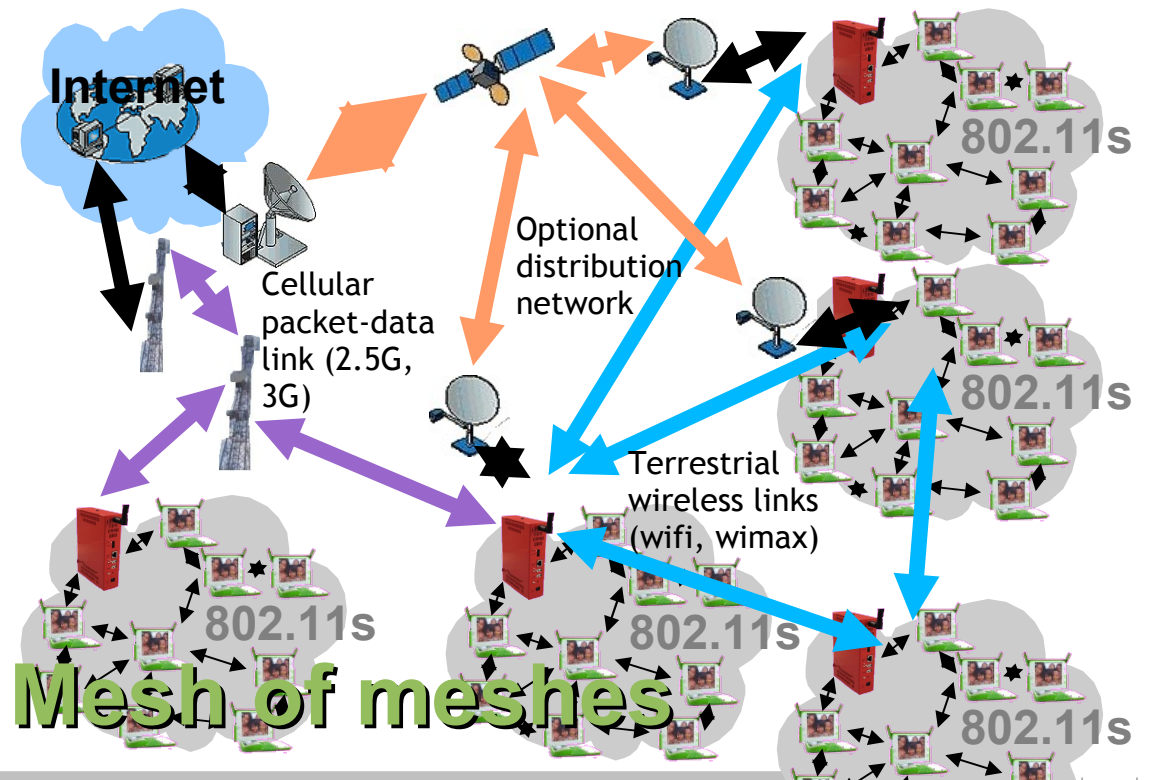
Mesh Networking

- Great technology!
- We need help!

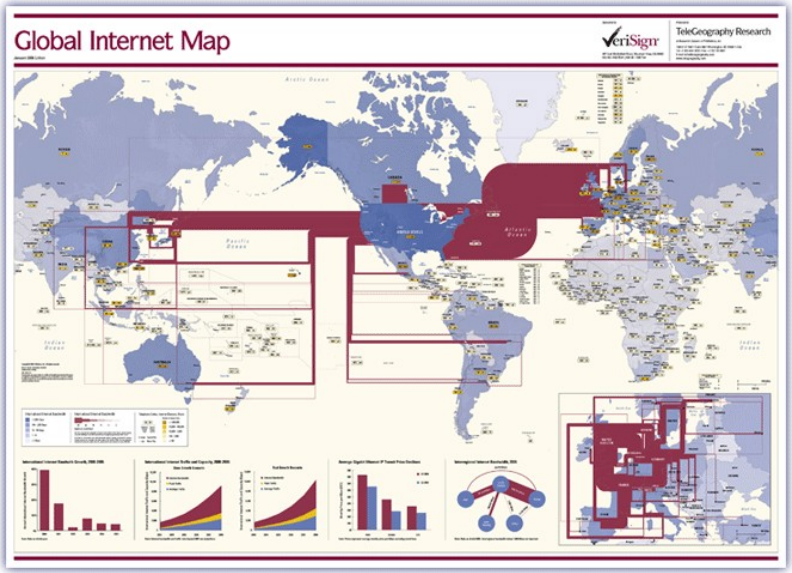
Connectivity

- Child to child communication is as important as child to Internet and child to teachers
- Wireless / Unlicensed / Build it Yourself
- Bandwidth is a perishable commodity


802.11s mesh


Internet
Cellular packet-data link (2.5G, 3G)
Optional distribution network
Terrestrial wireless links (wifi, wimax)
802.11s
802.11s
802.11s
802.11s
802.11s
Mesh of meshes

Not uniform, not global.



Build it yourself!


MAYA TURBO
HIND MOTORS
SUZUKI
JK 10 - 1777

But wait!

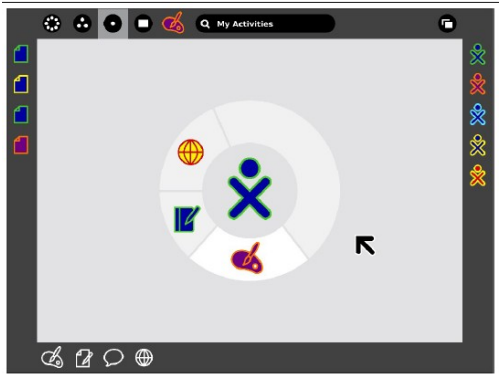
- Problem: no one else is using this!
  - Slogging through bugs in the 802.11s standard the hard & slow way
  - No one out there to help us write drivers and applications.
- One step towards solution: softmac driver
  - No more power benefit on XO =(
  - You can now help!

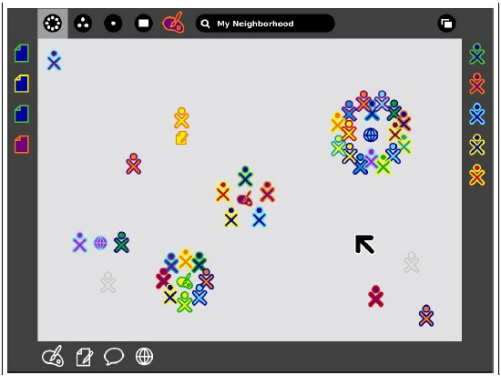
Sugar

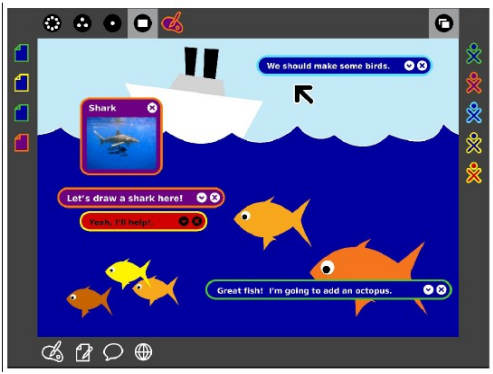
- Kid-friendly UI
- Discoverable
- Collaborative

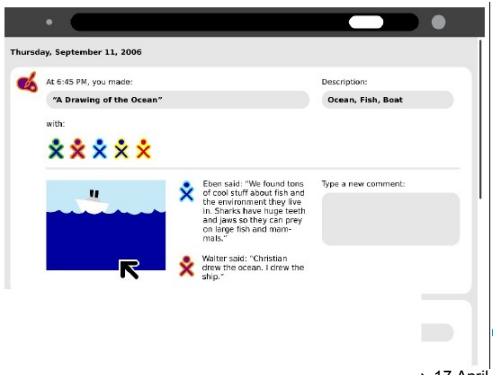
Zoom Interface

Peer to Peer User Interface









The perils of reinvention

- Sugar was a clean-slate reinvention
  - But kept X, gtk, dbus, python, pango, cairo...
  - Naively thought we could “sugarize” all apps.
- Integration problems:
  - UI toolkit written in python; hard to integrate with legacy C code
  - “Journal” only accessible via new interface
  - Blackbox wm not so sharp
  - Activities live in /home, not /usr/bin!

Goal: robustly support “legacy” apps

- Better wm/theme integration
- Journal is filesystem (next slides)
- Wrappers & tricks for security
- Package manager pieces???

Bitfrost security (in brief)

- Protect the user from applications
- “Click on everything” principle
- Original implementation using vserver
- New implementation using per-process UIDs and python wrapper
- Goal: port to Plain Old Linux
  - X server security work needed
  - Distro integration: how does this work?

Power Management

- Currently using ohm
  - Mostly userland
- Problems with timeouts!
- Move to sleepy-linux and cpuidle frameworks
  - Integrate with kernel scheduler: the kernel knows when it has to wake up next!
- Is this useful to anyone else?

Datastore: olpcfs

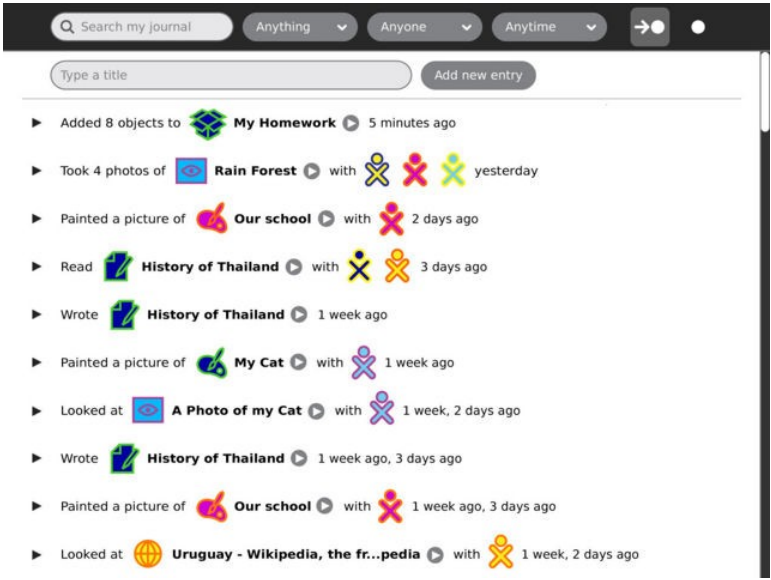
- Backing storage for the journal
- Original implementation used Python API
- Hard to work with!
- But this is what a filesystem does...

End-user goals

- Support journal and bulletin board abstractions
- Provide Bitfrost P_SF_RUN and P_SF_CORE protections

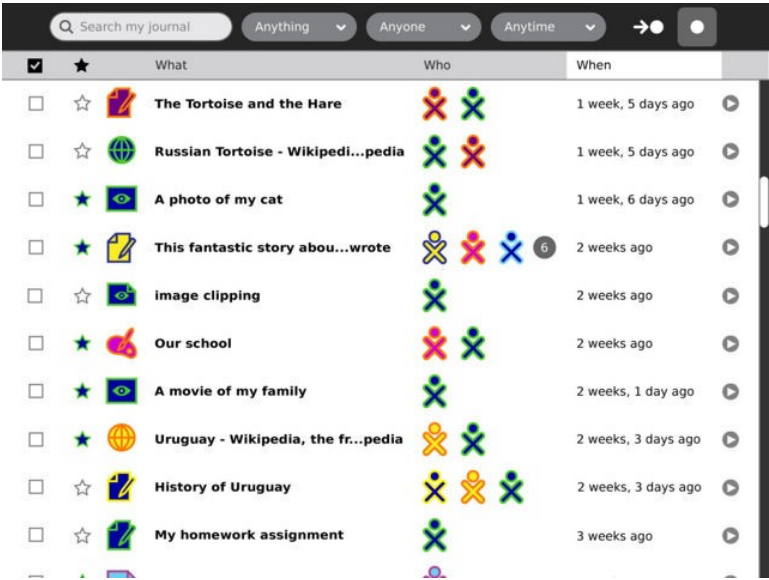
Journal: objects & actions

- Action view



Journal: objects & actions

- Object view



Design goals

- Filesystem w/ POSIX semantics
  - This is the codeword for “standard filesystem”. Windows, UNIX, and MacOS in various flavors have (more-or-less) POSIX-compliant filesystems.
  - Our first generation design was a “simple” proprietary wrapper: let's move forward!
  - Aim to provide ***best possible*** support for legacy applications

Design goals

- Content-addressable
  - Lots of attempts out there to create global distributed filesystems with unified namespaces – *let's not try this!*
  - Local arrangement & organization of documents is up to the individual user; all we need is an opaque tag to call it by.
  - Commercial support: XAM/Honeycomb (Sun)/Jackrabbit (Apache), etc, etc.

Design goals

- Versioned
  - Support exploratory learning by always allowing user to undo his most recent mistake.
    - “Continuous” versioning.
    - Snapshots don't work for this.
  - Groups of files may have independently modifiable *tagged versions* (“full persistence”)
    - Gives us our P_SF_CORE/P_SF_RUN support
    - Also very useful when importing collaborative work

The olpcfs *filesystem*

- Transparent **versioning**
  - Reach back and study the past – then change it!
- Rich **metadata** via POSIX xattrs
  - Enhanced by mechanisms to treat metadata as 1st-class files
- Integrated metadata **indexing**
  - Unifies “Journal” and “files & folders” views

Demo

- http://wiki.laptop.org/go/Olpcfs has pointers to the source
- 2,500 lines of Python
  - Bdb and python-fuse packages
- First impressions:
  - I prefer managing directory objects, rather than being given full pathnames

Journal integration

- All documents live in `~olpc/Documents`
- Sugar-aware activities add `action_id` xattrs for file grouping
- Add'l journal properties are directly implemented as xattrs

Journal, cont.

- Journal search built on native indexing
- Journal versions built on native versions
  - But additional attributes may be used for richer merge semantics, etc.
  - “Keep stars” in Journal correspond to landmark versions

Sync'ing & sharing

- All objects have “XUID”
  - Content-addressable
- Distributed indexes of various scopes on top of local index
- Not all local objects may appear in filesystem tree!
  - Some may be imported into index only

Implementation scope

- Lots of fallback/alternative implementations possible
  - Currently writing proof-of-concept to test APIs and unblock journal & other work
  - Non-versioned implementations with look-aside metadata easy using FUSE, lufs, 9P, etc.
- Flash filesystems are hard.
  - But the datastructures used here are very flash-friendly!

Conclusions

- OLPC's got lots of software work still left to do
  - And we need you!
  - And we need to make it useful to you!
- Please help:
  - Get mesh networking working! (and write cool apps)
  - “Sugarize” Edubuntu / Edubuntu-ize Sugar
  - Secure the desktop
  - Suspend often! (but wake up, too)
  - Write the Filesystem To Rule Them All

Questions?

- OLPC mailing list: devel@laptop.org
- Or ask me: cscott@laptop.org

The graveyard of unused slides.

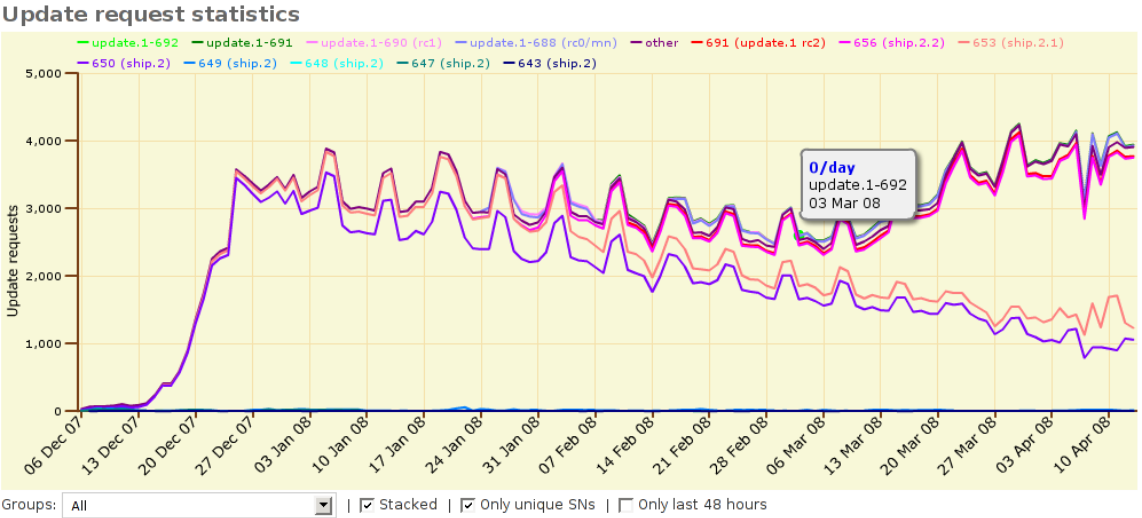
An expression machine

1. appropriate;

2. debug;

3. collaborate and critique.

Update server statistics



Bitfrost

- P_SF_CORE: no system files may be modified
- P_SF_RUN: the “working copies” of the system can't be modified

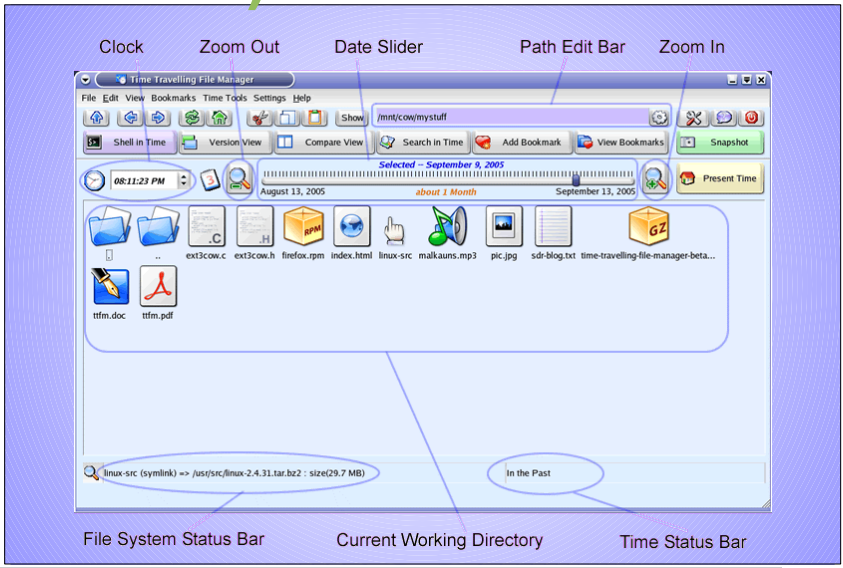
Bitfrost: it gets interesting!

- When #P_SF_RUN is disengaged... [i]nstead of loading the stored files directly, a COW (copy on write) image is constructed from them, and system files from that image are initialized as the running system. ... These modifications persist between boots, but only apply to the COW copies: the underlying system files remain untouched.

Bitfrost: turning P_SF_RUN back on.

- If #P_SF_RUN is re-engaged after being disabled, the boot-time loading of system files changes again; the system files are loaded into memory directly with no intermediate COW image, and marked read-only.
- End result:
  - Hacking is safe again!

Time-travelling file manager (an aside)



Sync'ing & sharing

- XUID encapsulates *object plus metadata*
  - “Who's got this XUID?”
  - “I'll tell you which XUIDs I don't have if you'll tell me your XUIDs.”
- Independently-modified documents may result in tagged versions in filesystem after import

Olpcfs directions

- Even if it's not as ambitious as this, our datastore should look like a filesystem!
- Lots to learn from BeOS & the BSDs; they rock!
  - Even NTFS