The Schools' Server: the Community XS

Sridhar Dhanapalan
Engineering Manager, OLPC Australia
2012-09-07

Contents:

Introduction

This is a high-level (not technical) white paper from OLPC Australia discussing the direction of the XS Schoolserver project. By empowering a more open, inclusive community of developers and users, XS development can be more responsive to needs on the ground and evolve more quickly.

Goal

The key requirement for the community XS project[1] is a flexible architecture that can suit a variety of purposes in deployments around the world. Deployments will be free to create their own products/distributions of the community XS project to suit their needs.

Through a combination of a flexible design and a professional approach, we aim to create critical mass in XS development through bringing together the various bespoke implementations into one sustainable project. This will give deployments greater certainty about the future viability of their installations.

About OLPC Australia

OLPC Australia believe strongly in providing a comprehensive educational programme. Technology creates new and interesting opportunities, but real education requires much more. We have created our One Education programme to deliver a holistic, sustainable solution to schools.

Professionalism is key. The core OLPC Australia team are employed full-time to focus on delivering complete outcomes.

All elements of the One Education programme are designed to closely interoperate. For Engineering, this means that we are in constant contact with other parts of the business such as Education, Support and Logistics.

Core Principles

We subscribe to OLPC's five core principles:

1. Child Ownership
2. Low Ages
3. Saturation
4. Connection
5. Free and Open Source

Beyond this, we have added two more:

6. **Empowering Teachers**
   A commitment to creating an environment that engages teachers as much as students is crucial to ensuring the ongoing educational benefit to students. Empowered teachers will be more inclined to take full advantage of the XOs to engage their students. This means working with the custodians of today's teachers – education departments and of tomorrow's teachers – universities.
7. **Community Engagement**

[1] Project vs Distribution vs Product

We believe that both Indigenous and Western ways of learning are equally important, and understand that children receive a great deal of their education in the home and community context. By consulting respectfully with local Elders, working closely with the community and grass-roots partners, and providing laptops to the community as well as to the children, we provide a flexible learning platform that parents and other interested community members can use to engage in innovative ways to help bridge the transfer of knowledge between generations.

Engineering Principles

The mission of the OLPC Australia Engineering team is to develop and deploy sustainable technologies that feel like an integral and transparent part of the educational experience. We want computing technology to be the pencil and paper of the 21st century.

The key is to engender *trust* in the technology. People on the ground must find the solutions to be reliable and dependable, not just to begin with but also over the long term. Teachers must be able to develop lesson plans using the technology and have full *confidence* that they will work in their classroom setting, even over time and across software updates.

We design for ‘normal people’ first. The user experience must be carefully managed to provide consistency between hardware/software and across versions.

We only deploy technologies that require *no* technical expertise to install or manage. Many technologies in the OLPC/Sugar ecosystem have an implicit assumption that technical expertise is somehow available. This creates a dependence on technical resources who may be scarce, unreliable or untrained in the technology in question. The reality often is that, for all practical purposes, there is no technical expertise available at all.

By assuming that *zero* technical expertise is available, we gain much greater clarity in design direction. When combined with further analysis of our deployments, we have assembled a framework to guide our solutions development:

1. It must ‘just work’ for a non-technical user, with only minimal instruction/training required.
2. If it is not as easy and reliable as using books, pencils and paper, the technology will not be sustainable.
3. If it requires significant (re)training, uptake of the technology will be limited.
4. Manage complexity with good design. User interfaces should be clean and simple to use. Provide a clear path for frequent actions.
5. At the same time, do not lock down the underlying platform. Allow for learning and innovation.
6. A simple ‘factory reset’ function provides a safety-net if something goes awry, empowering the user to experiment and push boundaries with confidence. We learn most from our mistakes.
7. Switching costs are high. A proposed solution must appear *significantly* better to the end user than the incumbent implementation, in order to justify the risk of migrating.

8. Attention to detail and consistency in the user experience and documentation can make a huge difference.
9. Instructional material (documentation, images, videos, etc.) should be kept up-to-date, in sync with the latest GA releases of the software and hardware.
10. Change control processes are used to manage user expectations between versions. Disruptive changes are only introduced with major releases.
11. Respect the role of the teacher. Teachers already have many responsibilities and demands placed upon them, and so they cannot be expected to be IT admins.
12. The One Education programme is focused on the teacher (class-based). Hence, the technology must be fully installable and manageable by a teacher.
13. Teacher turnover is high (only 8 months in some places), so it must be easy for a new teacher to start using the technology.
14. Truancy and mobility amongst children is high, so systems that require manual updating like registration and roster groups (presence segregation) are not practical.
15. Empower children and local community to ease the burden from the teacher. Parallelise technical support and maintenance tasks.
16. Software updates should be easy to manage in online and offline modes.
17. Network topologies differ, so we must be flexible. Many use WPA Enterprise with different credentials per person (even small children). Many use automatic proxy configuration (PAC/WPAD) scripts that require authentication on first use. Many place each client into a sandbox, preventing collaboration.
18. While our grass-roots deployment methodology has allowed us to gain traction in classrooms by bypassing bureaucracies, the downside is that we often do not have the support of network administrators. This means that we often have little knowledge or control over the schools' networks. We must assume that we are very limited in our ability to set up or change anything on the network.
19. Consider the impact on total cost of ownership for every change. 'Cost' is far more than the initial price of the hardware. The time and effort spent by people on the ground must be valued. 1 hour of engineering can save a collective 10,000 hours across all of our schools.
20. We often only have a short window of opportunity at the beginning to convince stakeholders to invest in the programme. To get our foot in the door, the solutions should show some immediate/quick potential. The Education market is full of expensive but well made/marketed solutions. We can learn from those examples.

OLPC Australia's Commitment

OLPC Australia commits to providing professional leadership of the new, community XS project. This means that we are prepared to be responsible for:

1. architectural direction
2. project management and issues tracking
3. source code management

In managing this project, we will be leveraging our experience from schools and communities involved in our One Education programme. The geographical scope of this

initiative is growing, and currently includes deployments across Australia, New Zealand, Papua New Guinea, Thailand and Oceania. That means that it must be congruous with factors that affect deployments such as supply chains, teacher training, marketing/branding and so on. We need to think comprehensively and maintain a careful change control process.

Through developing in this holistic context, we avoid delivering a technical system that is independent of deployment realities, or with exclusive focus on one deployment. Our expertise in product development and building usable systems that scale will be of great value to this project.

The design will be flexible enough to take into account the different needs and situations across deployments around the world. Complexity will be stripped down to its bare essentials (collaboration), with all other functionality available as installable add-ons. We are using standard GNU/Linux technologies to make this possible.

Through this flexibility, deployments can create their own products/distributions of the community XS as best suits them. There should be no need to create a fork of the project.

Historical Context

We owe a debt of gratitude to OLPC for creating the **X**O School **S**erver (XS). The XS enhances the XO experience in a school through providing local network and Internet services.

Sadly, in recent years the XS project has not received adequate resources for sustained development. Releases have been sporadic and have lagged noticeably behind the available technology. This is in part a consequence of being a fork of Fedora.

The response from the community has been to fork the project into several directions. This has resulted in increasing fragmentation, with at least seven variants presently in existence. While each may presently work well for its target deployment scenario, many are one-off forks.

Such implementations are not respectful of the ongoing needs of the deployment. It will create maintenance problems as time goes on and deployments' needs evolve. It certainly does not promote sustainability on the part of technical development or local community.

The XS development community is too small to sustain multiple bespoke implementations in parallel. The solution is a flexible architecture that can be configured to suit a deployment's needs without the need for forking.

In our XS-AU (currently, One Network) project, now over two years old, we pioneered the idea of a simplified, modular set of software that could be easily installed on top of an existing installation of Fedora. The architecture of XS 0.7 is a vindication of that effort. We

are keen to continue that work and continue to innovate, working with the global community so that deployments worldwide may benefit.

This is an opportunity for the community to continue the good work that OLPC started, and assume a greater share of the load and responsibility. Through sustained development and a mind on the needs of deployments, we can bring the project to the next level of maturity.

Community Building

A community XS, by definition, needs a community in order to be sustainable. OLPC Australia can provide overall direction and lend experience to the project. However, it is up to deployments to participate and ensure that they are adequately represented in the development process.

In particular, a wide testing base will better guarantee success. We would like to see the community XS trialled in a variety of scenarios, with full feedback being fed back into development. This may include data collection, analysis and publication of results. This will enable releases to be improved upon based upon lessons learned from the field.

Project vs Distribution vs Product

It is important to define the concepts of project, distribution and product.

A *project* is an ongoing effort to develop a technical solution. It is under constant flux and hence is not generally intended for end-users. A *distribution* is a packaging of a version/snapshot of the project's files, often configured for a particular purpose.

A *product* is more comprehensive, providing properties such as ease of use, QA and accompanying documentation. What makes a product really stand out is integration with other parts of the deployment organisation's business. This can include factors such as supply chain, technical support and the educational programme.

The community XS is a project, open to participation/use by anyone. Deployments are encouraged to create their own distribution/product to better suit their needs. For OLPC Australia, this will take the form of our One Network product.

Professionalism

Professionalism is what makes a community project into a great product. Through being a social enterprise and not just a charity, OLPC Australia have been able to build a network of partners and sponsors across a range of industries, including banking and financial services, telecommunications, media, marketing and information technology.

Extremely important to us is the creation of products and processes that scale, across their

entire supply chain. Our schools must be backed by technologies that have longevity and sustainability. The creation of a finished product will mean that we have something that feels like a natural extension to our programme, and hence will feel inviting to schools and communities.

Of course, the end-user product is underpinned by the community project. We need effective leadership and scalability in processes and team.

The combination of a flexible architecture and a professional approach means that we can accommodate a variety of end-user scenarios while creating sustainability for deployments.

Marketing and Positioning

An important but oft-overlooked factor in free and open source software projects is marketing. This is not a dirty word, as some may think. The fact is that effective communications around a product can lead to a better perception and uptake. This is particularly important when the target audience is different from the people who are developing the project.

Effective marketing starts at the project level, but can become critical at the product level.

For example, one idea we are examining is to drop technical-sounding concepts like “server” from our nomenclature and language. We have an opportunity to integrate server functionality into Sugar in a seamless, inviting way.

We may want to reconsider the name of our project before we open it to all.

Community XS Design

This section outlines the architecture of the community XS.

Scenarios

Here are some scenarios that the community XS should be compatible with. Non-conflicting combinations of these should also be possible.

1. it hosts the network, acting as an access point itself
2. it hosts the network, and bridges to an external wireless access point
3. it acts as a gateway to another network, such as the Internet
4. it participates on an existing network, without hosting core services
5. one XS server serves the whole school
6. many XS servers participate on the same network
7. the XS optionally provides services such as collaboration, registration, roster groups (presence segregation), backups and content on a modular basis

Requirements

The community XS project aims to be flexible enough to satisfy a variety of needs in local deployments. Some groups may go further and create their own distribution or product to suit a particular scenario.

Basic requirements include:

1. the project must be easy to maintain by developers, recognising that we are a small community
2. the development environment must run a standard desktop system, e.g. in a virtual machine
3. use proven technologies where available - hardware and software components and projects used by the community XS must themselves be well-established, well-maintained and have scalability in development and capability
4. it should be easy to create distributions/products without needing to fork the project
5. products built from the project must be easy to install and maintain
6. can be installed in headless mode from USB
7. Linux sysadmins should be able to easily understand the setup
8. compatible with a range of different hardware
9. works on x86-32, x86-64 and ARMv7 architectures
10. scalable down to low-cost hardware, including an XO-1.75
11. scalable up to powerful hardware, to handle more clients/functionality
12. able to be stripped down to very basic functionality (collaboration services only)
13. additional capabilities can be easily added as needed
14. features can be controlled via a Web interface
15. services which were previously required such as idmgr (registration) and Moodle are now optional
16. additional storage and content on external media can be easily made available to clients
17. multiple servers can be run on the same network (e.g. one per classroom)
18. plays well on existing networks, with only one interface required
19. can optionally run core network services, such as DNS and DHCP
20. can optionally (with the right hardware) create its own wireless network for client XOs to connect
21. can optionally provide gateway services, e.g. for Internet access
22. can provide Internet access for clients via USB 3G modem
23. general functionality can be controlled by a non-technical user in a friendly, graphical way (only minimal training required)
24. for more complex functionality, is easy to maintain by someone familiar with Linux.
25. potential for centralised monitoring, data collection and management, e.g. via SSH or OpenVPN
26. potential for integration onto the OLPC OS running on an XO
27. potential for integration with Sugar
28. clients should be able to easily find the correct server on the network

Implementation

A community XS should be implemented as a set of packages (maintained in a yum repo) that install on top of *Fedora 17*. This gives us several advantages:

1. we automatically leverage the properties of Fedora (features, packages availability, QA, hardware compatibility, etc.)
2. by not forking the OS, we are compatible with the expectations and training of people who are already used to Fedora/RHEL/CentOS
3. it can work on x86-32, x86-64 and ARMv7
4. it can be installed on small devices that are F17 compatible, such as a GuruPlug
5. it is compatible with the OLPC OS (which is based on Fedora 17), and hence can be installed on an XO as an add-on
6. upwards scalability can be achieved by installing onto Fedora running on different hardware, e.g. as a yum repo or as a Fedora spin
7. additional XS/Fedora features can be installed with yum/rpm

By being installable on the OLPC OS on an XO as an add-on, we gain further advantages:

1. we are building upon the familiarity, training and knowledge that the teacher already has, rather than pushing something foreign upon them
2. we have a platform that is widely available, affordable, rugged and repairable
3. power efficiency, and can even run off-grid with battery and solar
4. reliance on a separate chain for hardware supply, development, QA and technical support is eliminated, slashing costs for a deployment
5. installation on an XO can be easy, e.g. via bootable USB Customisation Stick or network install with yum
6. we inherit tight integration with the hardware at the OS level (e.g. power management and watchdog)
7. we benefit from the ongoing development of the OLPC OS
8. we can incorporate the capabilities and configuration tools into Sugar (e.g. as Control Panel applets)
9. we can use the server functionality in tandem with existing/new Sugar functionality (e.g. Transfer to Many)

One Network

One Network is the current working title of the networking solution that is being worked on by OLPC Australia. It includes a server product based on the community XS technology.

In addition to fulfilling the needs of the One Education programme, the One Network server may act as a reference to deployments of a product that can be built using the community XS project base.

Consistent with the teacher-centric approach of One Education, One Network is engineered to operate at a classroom level (rather than a school-level). Hence, we want to make it simple and sustainable for each teacher to set up and manage their own server and

networking for their classroom. Our solution is very turnkey and appliance-like.

Implicit in our design is the recognition that teachers are very time-poor and generally have no inclination to be IT technicians/admins. The technology should be a net time saver, so that the teacher can educate more effectively than through traditional methods.

Requirements

The platform for the One Network server is an ARMv7-based XO, running the One Education OS (based on OLPC OS). This makes development, and deployment and support far simpler than a standalone distribution. The OS can be extended with server capabilities using a bootable USB Customisation Stick (offline) or yum.

Deployment requirements:

1. Requires *no* technical expertise. It should be easy and quick to deploy by a totally non-technical user.
2. Works within the teacher's mental model. Leverages existing knowledge rather than requiring retraining.
3. Is compatible with our classroom-level (not school-level) education programme. Each certified teacher should be able to run their own server on the network.
4. Affordable TCO.
5. Minimises supply chain, maintenance, QA and technical support issues.

Technical requirements:

1. Easily install on top of One Education OS using USB Customisation Stick or yum.
2. Easily install additional features using USB Customisation Stick or yum.
3. Easily update via USB Customisation Stick or yum.
4. Capable of handling at least 30 clients.
5. Start with basic collaboration server functionality only.
6. Complex features like registration, Moodle and roster groups (presence segregation) are not included by default (but may be installed).
7. A Sugar Control Panel applet shows available services to activate/deactivate, based on what is installed.
8. Must be compatible with the different network topologies found in One Education schools, and not require changes to the network.
9. Only manage the network when necessary.
10. Directly handle Internet connections, if required.
11. Offer Internet gateway services via WiFi, 3G/4G or USB2Ethernet adapter.
12. Minimal dependence on existing resources on the network.
13. Can be easily expanded with external storage.
14. The correct server be easily found on the network by children's XOs, using mDNS (Avahi).
15. For developers, the project should be easy to maintain.

User Stories

User story for a teacher ("Mrs Mac"):

1. Mrs Mac pulls their new XO out of the box and turns it on. She can see that it is running the OLPC OS.
2. She shuts down the XO, inserts her One Network server Customisation Stick and then turns it on again
3. The XO boots from the USB stick and automatically installs the One Network server packages.
4. The process ends with a friendly message explaining that the installation is complete.
5. She shuts down the XO, removes the stick and turns it on again.
6. The XO boots as normal.
7. In the Neighbourhood view, Mrs Mac connects to the school's wireless network.
8. She goes to My Settings and sees a new applet: *Services*.
9. In this applet, there are tickboxes to toggle services on and off. This list is dynamically updated depending on what is installed on the XO. By default there should only be one: *Collaboration*.
10. She ticks the box and then clicks the big tick icon in the top-right-hand corner of the applet to exit. This starts ejabberd in the background, which advertises its presence on the network just as a printer would (using mDNS with Avahi).

User story for a child ("Geoffrey"):

1. Geoffrey turns on his brand-new XO.
2. In Sugar, he goes to the Neighbourhood view.
3. He connects to the school's wireless network.
4. Geoffrey sees that Mrs Mac's collaboration server is displayed as a distinct server icon (using Mrs Mac's colours) in the Neighbourhood view.
5. Geoffrey clicks on the icon. His XO connects to that collaboration server.
6. Now, all of Geoffrey's collaboration traffic travels via the server (telepathy-gabble).
7. Any other services available on the server are also made available to Geoffrey.
8. An icon in the frame indicates that Geoffrey is connected to the server. The same icon can be used to disconnect and switch back to using link-local (telepathy-salut).