Report of the Layered Athena Team

Bruce Lewis - Team Leader

June 1, 1995

Contents

1 Introduction

In the fall of 1992, DCNS Development initiated the Layered Athena project, with two major motivations: To make it easier for private workstation owners to use Athena, and to make it easier to port Athena to new platforms. Of these two, the hope of easier Athena ports was probably the motivation for management to devote key resources to the project. Another goal of that fell out of the above goals is actually identifying what Athena is.

This report details the extent to which Layered Athena has met these goals, and serves as compilation of knowledge that has been gathered from the efforts of the Layered Athena team, most recently Kevin Cunningham, Bruce Lewis, John Morey, and Naomi Schmidt. We owe much of the knowledge herein to people who have been active on the team, particularly Carla Fermann and Mark Rosenstein.

1.1 Summary of Recommendations

Ports to New Platforms

When Athena is ported to a new platform, Layered Athena should be ported immediately after the first system packs. The new platform should be supplied to ACS, CSS and SIPB, who can use the xlayer GUI to take updates as work progresses on the port. This way, work can begin on courseware, utilities and other locker software early on.

When a vendor introduces a new OS version with binary compatibility, a workstation should be installed with the new version ASAP, and Layered Athena should be used to make it into an Athena workstation. This will help potential problems with the new OS version be identified quickly.

Layered Athena for Private Workstations

I/S should not attempt to sell the MIT community on the benefits of Layered Athena. We should wait for customers to approach us with needs that Layered Athena could meet, and take appropriate action based on the level of effort involved versus the potential benefit to the MIT community.

We should work with these customers to help them (1) understand the lack of support so that they can avoid too much reliance on the system, and (2) convey this understanding to their users. For example, a faculty member with a special cluster needs to understand that if any coursework depends on Layered Athena, there must be a contingency plan in case it suddenly stops working. We would need to work with such a faculty member to develop a handout for students that would warn them not to rely too heavily on the system continuing to work, and tell them what level of OLC support to expect.

Conceptual Models of Athena

When a "new model" for Athena is proposed, it should be described in terms of the three components described in this document: system packs, installation software and locker software. It should be pointed out where the new model is different. When there are differences in the system packs, Figure 1 should be used to illustrate which subsets are different. This will demystify the developer's perspective of the new model.

Of course, the new model should also be described from the user's perspective, i.e. what the workstation looks like when you login. A straightforward change from the developer's perspective may result in a radically different model from the user's perspective, and vice versa.

2 Layered Athena Can Expedite Ports

At the beginning of the Layered Athena project, it was hoped that a new model for software installation could replace the current model, which is development-intensive and time-consuming. The Layered Athena team has come to understand that the current model for installation is not yet ready to be replaced, because Layered Athena does not meet the requirements detailed in section 4.2. However, Layered Athena is still useful as an interim installation procedure before machines are installed in public clusters.

A timeline for a non-layered port can look something like this:

```
                                            cluster deployment
DCNS dev:                                   |
srvd    aabbccbcccbccccbcccccbccccccbcccccc|bbbccc
inst1   ---------aaaabbbbcbccbcccbccccbcc--|
inst2   -------------------------aaaabbbbcc|dd
Non-DCNS dev:                               |
srvd    ---------------------cbbbccbcbccbcc|cbcccc
lockers ---------------------------aabbaabb|aabbaa
```

In this timeline, the symbol (-) means nothing is done, (a) means development work, (b) means finding and fixing bugs, (c) means use, and (d) means documentation. The port starts with developers doing the simplest task, which is building system packs (srvd). After the srvd has undergone some developer testing, it needs to be installed on several non-developer machines for further testing, and so that key personnel can familiarize themselves with the new platform.

To install these machines, it isn't necessary to have a procedure that fits all of the cluster group's requirements, so an intermediate procedure is developed (inst1). Depending on what roadblocks are encountered, this procedure can take as long to develop as the final install procedure. It may need handholding, meaning a developer needs to be involved in every installation. This can take a significant amount of developer time away from debugging the srvd and writing the final install procedure.

As the time approaches to put machines in the clusters, the final install procedure (inst2) must be written. Because of the load on developer time up to this point, inst2 is saved until the last minute, and finished just in time.

Courseware, applications and tools in various lockers need to be compiled on the new platform. Unfortunately, groups such as ACS, CSS and SIPB don't get the srvd software until relatively late in the game, and must scramble to get everything working. It's expected that a lot of locker software just won't be ready until well after the new machines are deployed in clusters.

Layered Athena makes possible a totally new timeline:

```
                                            cluster deployment
DCNS dev:                                   |
srvd    aabbccbccbbcccbbccccbbcccccccccccc|cccccc
layer   ----abc----c----c-----c------------|
inst2   ----------aaaabbbbccdd-------------|
Non-DCNS dev:                               |
layer   -------c---c----c-----c------------|
srvd    -------cbbbccbcbccbccccbcccccccccc|
lockers -------------aabbaabbaabbaabbaabbbc|bcbccbccc
```

After the initial srvd has been built and lockers can be attached, porting Layered Athena to a new platform takes less than a week. (Later, when a vendor introduces a new OS version with binary compatibility, the Layered Athena software developed for the previous version should continue to work with little or no modification.)

New machines other than the ones developers are using should be delivered with the native operating system pre-installed. The Layered Athena GUI can then be run without any developer assistance to install srvd software, and to easily take updates as they become available.

As a result, the srvd gets more testing sooner, developers are freed up to work on the final install procedure, and work on locker software can begin long before machines hit the clusters.

The initial srvd need not be complete. Login, the most difficult part of the srvd, is not essential for private workstations. The native login can be used, along with a script called `login_athena` which includes all the elements of an Athena tty login. Users get their usual prompt, shell, PATH, aliases, etc., and can port locker software in the environment to which they have become accustomed.

2.1 Recommendations

When Athena is ported to a new platform, Layered Athena should be ported immediately after the first system packs. The new platform should be supplied to ACS, CSS and SIPB, who can use the xlayer GUI to take updates as work progresses on the port. This way, work can begin on courseware, utilities and other locker software early on.

When a vendor introduces a new OS version with binary compatibility, a workstation should be installed with the new version ASAP, and Layered Athena should be used to make it into an Athena workstation. This will help potential problems with the new OS version be identified quickly.

3 Don't Promote Layered Athena at MIT

Over the years, Athena has established high expectations in the user community regarding support. In the Layered Athena world, it becomes difficult to meet these expectations, especially when the platform is not found in Athena clusters or in offices of user support groups. Setting realistic expectations is of major importance whenever Layered Athena is deployed.

Support groups need to be made aware of any deployment of Layered Athena, and will need a plan to properly handle the inevitable questions.

Some headline issues the Layered Athena team encountered are:

- How do we set realistic expectations?
- How do we prepare and coordinate those who directly support testers?
- machines and tools for adequate support
- feedback / data collection
- timing / scheduling
- proactive support (documentation, etc.)
- vendor support in a layered world

It should be noted that the beta testers of Layered Athena did not demand much support, but they are probably not a sufficiently random sample of the MIT community for us to conclude that others will be like them. People who agree to a beta test tend to have lower expectations, and to be more knowledgeable. The six principal non-I/S beta testers were all graduate students or staff.

3.1 Recommendations

I/S should not attempt to sell the MIT community on the benefits of Layered Athena. We should wait for customers to approach us with needs that Layered Athena could meet, and take appropriate action based on the level of effort involved versus the potential benefit to the MIT community.

We should work with these customers to help them (1) understand the lack of support so that they can avoid too much reliance on the system, and (2) convey this understanding to their users. For example, a faculty member with a special cluster needs to understand that if any coursework depends on Layered Athena, there must be a contingency plan in case it suddenly stops working. We would need to work with such a faculty member to develop a handout for students that would warn them not to rely too heavily on the system continuing to work, and tell them what level of OLC support to expect.

4 Athena's Components Demystified

The Athena environment is a very complex entity. Any attempt to completely define it is likely to be full of gaps. For example, one might forget the network services that must be available, or the knowledge that must exist in support groups for customers to get the kind of service to which they have become accustomed.

However, the work done in Layered Athena makes it possible to answer the question, "What software makes a Unix workstation into an Athena workstation?" A similar question is, "What development work needs to be done for an Athena port?"

These questions become very important when we begin to look into new models for the Athena computing environment. A picture of the components of this environment make it easy to point out how a new model is and isn't different from our present model, thus making it easier to estimate how much more or less effort will be involved in the "new model" port.

An Athena port has three main software components:

1. Athena software on system packs, i.e. /srvd
2. Install - software to install an Athena workstation
3. Locker software

Layered Athena deals primarily with the anatomy of the system packs, but the other components are described here for completeness. Forgetting the about installation process is a grave error if one wishes to put a new Athena platform in public clusters. Forgetting locker software shows that one is out of touch with the expectations of the Athena user community.

4.1 The System Packs

Layered Athena was designed to allow private workstation owners to choose which parts of the Athena systems packs they would like to install. To achieve this, it was necessary to come up with a division of the system pack into subsets that were meaningful to users, and to determine what subsets depended on what other subsets. Figure 1 illustrates this division.

Kerberos This subset installs or modifies the necessary configuration files for support of Kerberos v4 clients. It installs basic clients, `kinit, klist, kpasswd, rkinit, ksu`, and the utility programs `ksrvtgt` and `ksrvutil`. It does not require modifying system startup files. The Kerberos subset could easily be extended to include Kerberized applications such as telnet and rlogin, providing an easy-to-use solution for private workstation owners who want to protect their passwords, but don't want any other Athena software.

AFS This subset installs configuration files and the AFS cache directory, and modifies system startup files so that AFS is started on reboot. It also installs the utilities `aklog, unlog, fs, pts`, and `vos`. If a private workstation owner wants to get at files in AFS (which includes virtually all Athena lockers), the Kerberos and AFS subsets are enough.

Basic Athena Services Many Athena services are so simple and unintrusive that they can be packaged together in one subset, even though there's a wide variety of functionality. These services include hesiod, zephyr, moira, online consulting, global message of the day, kpop email, authenticated printing, and attach.

To ease use of these services when the login subset is not loaded, a script called `login_athena` is included which will allows the user to get tickets, Athena home directory, zephyr, and any other necessary setup.

X The Athena X subset includes the standard X11 clients as we compile them. For newer Unix platforms, this subset is unnecessary, as vendors have largely caught up with Athena in terms of X releases.

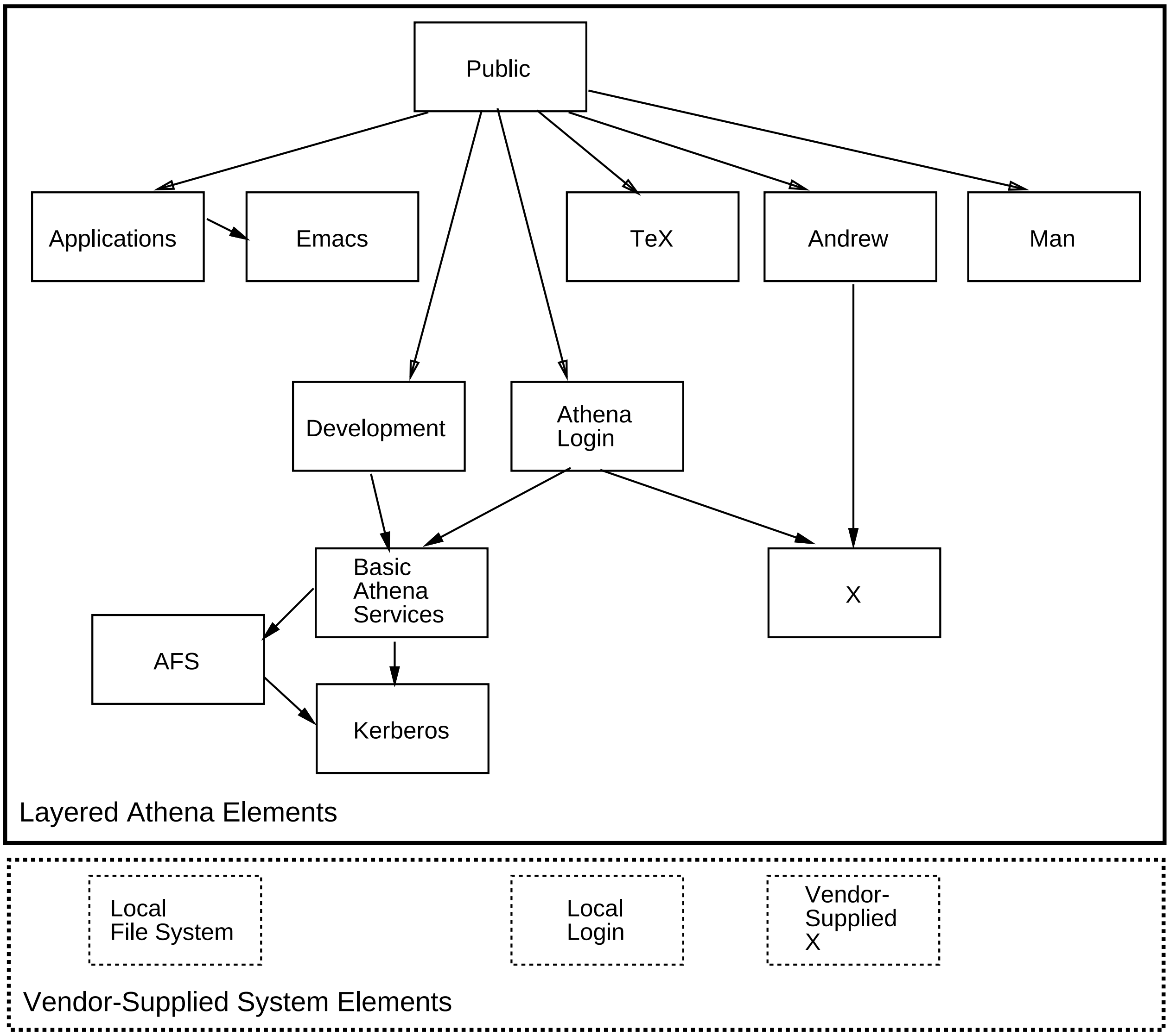

Figure 1: Subsets of the srvd with dependencies

Login This subset runs the Athena login program on the workstation. This means that anyone with an active Athena account can login to the workstation unless access is specifically restricted. This subset includes a lot of interaction with the base operating system, and is typically the most difficult part of compiling system packs for a new platform.

Emacs This subset includes `emacs` and its libraries and utilities. Many private workstation owners would like an easy way to install emacs without having to compile it themselves, even if they don't want any other Athena functionality.

Man This subset includes the man pages for all Athena programs and on some platforms, a modified `man` program as well. In the future, man pages may be bundled with the other subsets rather than be separate.

Tex This subset provides the TEX and LATEX formatters and associated utilities. Like emacs, it is a third-party package some private workstation owners may want, even if they don't want any other Athena functionality.

Andrew The subset provides `ez` and other applications built on the Andrew Tool Kit.

Applications This subset contains Athena applications that don't fit into the "Basic Athena Services" or other subsets. They are small, easily-ported programs, so that it's convenient to lump them together into one subset despite the wide range of functionality provided. Applications include: `olh, delete, mh, rcs, tcsh, eos, imake, notes`, kerberized remote access, `discuss, dash, perl`, and much more.

Development This subset contains the libraries, include files, and other tools necessary to write programs that use the Athena basic services.

Public This subset was never implemented. It would handle automatic updates to new releases, etc. It was hoped that one day Layered Athena would reduce the effort needed to create an install kit for public workstations, however, the current Layered Athena paradigm cannot satisfy the requirements for that installation process, because it does not reinstall the base operating system.

4.2 The Installation Process

As most readers of this document know, about 400 Athena workstations are maintained by a staff of 5. For this reason, the traditional Unix installation process is far too cumbersome. Not only do new workstations need to be installed en masse during a short period during the summer, but hacked or non-functional workstations need reinstallation. DCNS-cluster does roughly 140 reinstalls per semester.

A reinstall usually involves inserting the install media, typing a small number of configuration variables (e.g. IP address), waiting for the install media to pop out, and then letting the installation process continue over the network.

To keep this job manageable, the following times must be kept small:

1. Time until done typing. The native install procedure for most versions of Unix involves a lot of typing, and there's no way to parallelize this task without hiring more people. For current platforms, staff spend less than two minutes on this stage of a reinstall.

2. Time until the install media pops out. The sooner the install media pops out, the sooner the installation of another workstation can begin. For current platforms, this takes anywhere from 5 to 20 minutes. DEC 3100's, DEC 5000's and SGI's do not require install media; they do the entire installation off the net.

3. Time until the installation process finishes. If everything goes well, it's acceptable for this stage to go on for hours. However, when something goes wrong, it is necessary for a staff member to sit in front of the workstation and watch the install process happen to find out what's breaking. Current platforms take about 45 minutes to finish installing themselves.

Until these requirements change, any new public workstation platform will need an install procedure. This has always been one of the most difficult parts of Athena ports.

For more details on Athena installation, see the file `/mit/opdoc/install/software.install.overview` on Athena.

4.3 Locker Software

ACS and CSS keep track of what locker software is most important. See `/mit/release77/WhatShouldRun` on Athena for details. Users also expect software from a variety of lockers to be available, notably the sipb and graphics lockers. When porting to a new platform, it is advantageous to make it available to the SIPB as soon as possible.

4.4 Recommendations

When a "new model" for Athena is proposed, it should be described in terms of the three components described in this document: system packs, installation software and locker software. It should be pointed out where the new model is different. When there are differences in the system packs, Figure 1 should be used to illustrate which subsets are different. This will demystify the developer's perspective of the new model.

Of course, the new model should also be described from the user's perspective, i.e. what the workstation looks like when you login. A straightforward change from the developer's perspective may result in a radically different model from the user's perspective, and vice versa.