FLOSS in Academic Computing Infrastructure: Experiences in Computer Science, Mathematics and Informatics Departments

Mehmet Gençer[1] and Chris Stephenson[1]

İstanbul Bilgi University, Department of Computer Science
Kurtuluş Deresi Cad. No:47 34440 Dolapdere, İstanbul, Turkey
{mgencer,cs}@cs.bilgi.edu.tr

Abstract. This paper reviews experiences in and solutions for a computing infrastructure serving Computer Science, Mathematics and Business Informatics departments. Strategic choice of using open source was primarily motivated by lower costs and higher level of system integration the open source GNU/Linux systems provide. However use of OSS solutions has led to fortunate improvements in several other aspects which were not clearly predicted at the beginning. The overall improvements were: (1) a higher level of system integration, (2) lower maintenance costs, (3) lower hardware costs, (4) better experience in education, and (5) increased flexibility and capacity in team problem solving for system administration. Organizational politics aspects are also reviewed as our problem has been one of trying to move to FLOSS in an institution where decision making is not particularly transparent and entrenched interests are against change.

1 Introduction

The range of computing infrastructure needs of a Computer Science department tends to be wider than that of any other department and requires a departmental level of control over the computer systems. However, typically the provided computing infrastructure is developed and supported by an organization wide IT department whose target user prototype is considerably different from Computer Science students and staff. In this respect the management of computer systems in the University has traditionally been a field of struggle which is a mixture of practicality and politics.

Our problem has been one of trying to move to FLOSS in an institution where decision making is not particularly transparent and entrenched interests are against change.

This has involved a mixture of practical endeavor, politics and diplomacy. More than 5 years down the road we have some achievements but still much to do. In this short paper we try to draw some lessons from the experiences in our University.

2 Short history

Bilgi University Computer Science Department is a new department in a new University. The University was founded in 1996 and we took in our first students (we were then only a section of the Math department) in 1997.

In 1999, when the present Computer Science Department head asked how to upload some files for students on to the University's web site, he was told by the IT department,"Oh just send us the .doc files and they will be up on the site within a month or so".

Given there was no obvious immediate internal solution, the department simply started to use rented space owned by the department head, available on an external server. This at least established the principle of online course data for students.

Once this tradition had been established, over a couple of terms, it was time to try to move to become independent of the IT department in two respects: (1)to establish our own servers and (2)to gain control over our own computer labs. We acquired a third hand Pentium 233 from another department for our server, and, after a long period of negotiation, the necessary DNS entry and network connection. We also won acceptance that our practical classes would not just be allocated from the general pool of computer labs in the University, but that there would be two labs reserved to the CS department. This was essential if we were to start to change the software installed on those computers.

It was around this time that the department took the decision to set itself some strategic targets, and one of those was to take up and promote FLOSS software. It now seems incredible how unsure we were of ourselves in those days. We had very few students and no graduates yet.

FLOSS software was attractive for several reasons. Documentation for open source software is much more accessible compared to their proprietary counterparts. This situation would ease student requirements for self-study. Also they would be able to install these software in their home computers legally. In addition when instructors wish to use new software for their courses, these can be provided quickly -as no purchasing phase is necessary, not mentioning the immediate cost savings.

We set up our new server using Red Hat 7 and made all our lab machines dual boot. The server sat under the desk of the head of department. We also set ourselves two objectives: (a) uploading course material to the site should be very easy, and (b) students should see an immediate improvement in the service they got, in particular they should no longer have live in a diskette economy where they carried their files on diskettes and submitted their projects on diskettes.

The reason for (a) was that many of our full and part time teaching staff were surprisingly computer illiterate. If they were to use the system it had to be really easy; and for (b) was because we -correctly- anticipated resistance from students. We felt we needed a carrot to make the change to FLOSS more attractive.

After a number of failed attempts, the problem of easy upload was solved by mounting the web directories so academic users could simply save files to them. However this did not solve the problem of maintaining and updating links to new material on the system. This was solved with a three page PHP script, written over a few days by one person. This simply created a cascading menu system linking to any of a number of well used file types (pdf, html, txt) uploaded into the relevant directories. It was a zero maintenance, near to zero development cost primitive content management system. This was something that would only have been possible in a FLOSS environment. Despite the rather ugly interface, the system is still in use, virtually unaltered, today.

In the labs we made probably our biggest mistake. We made our labs Windows/Linux dual boot, and then tried to provide our improved computing environment to students in under both operating systems. This presented us with an development task we did not want and lacked the resources to do. We were only able to make progress when we dropped Windows and decided that we would only support Linux in our labs.

Once we had our server system up and running our next step was to acquire some more third hand computers and a switch, set up a gateway, a file server and mail services. We established mailman groups for all our courses and for the department. Finally, a move of campus two years later allowed us to make the case, showing what we had achieved with essentially free equipment, to finally spend some money getting a rack of new servers and our own independent Internet connection. The department now has 13 servers and around 200 desktops, all 100% FLOSS.

The only two items of software which incur any license cost in our system are a virus protection system for the few staff PCs that are still running Windows, and a Mat-lab system (which actually did not work despite the claimed compatibility and provider's all efforts).

3 Follow me Computing

Follow-Me-Computing was probably our best idea, but it was not easy to achieve. The idea was that we would virtualise student access to our computer resources. Anywhere the student went, a log in would be enough to get the student's own desktop set up, and personal files on the screen in front of them, whether they were in the University or at home. Our first mistake was to try to make a dual OS implementation. Providing the facilities we wanted under the restrictions of a proprietary operating system proved impossible. Only when we abandoned this course could we make progress.

With a basic Linux only system a high level of integration was achieved for users which we have failed to achieve with commercial operating systems. In what we call "follow-me-computing" environment, a user may work in any client computer by logging in with their password. The computer is integrated with the central home file-system via NFS(Network File System) so that the

user will be greeted with their desktop and language settings, and can access their personal files ubiquitously so that they do not have to carry their files around on other media or be upset with language settings on their desktop. In addition SSH(Secure SHell) enables users to access these files with relative ease when using a computer outside the campus. This second type of access is not limited to Linux computers.

3.1 Current practice and technology

Linux systems allowed us to boot client computers over the network using PXE technology, instead of booting from local hard disks. There are various advantages of such a system: (1) no hard disk drives are necessary, (2) all client software is updated once, thus reducing maintenance overload substantially in addition to savings on hardware costs.

4 Advantages to user and to management

The advantages of our current system, compared with the environment provided to users of the proprietary system outside the CS department, to both teaching and student users are fairly obvious:

- Ease of access
- Permanent, backed up storage
- Large mail accounts
- Easy to set up mail groups
- Instant loading of web pages

This is to say nothing of the obvious educational advantages of a FLOSS environment.

There are other advantages that are less obvious. While the proprietary environment outside of CS has a considerable staff involved full time in looking after the system, cleaning viruses, reinstalling software, the CS systems require no specialized staff at all. The maintenance load is extremely low, even though it may be irksome when it is carried by someone who has another, full time, job.

Another important advantage is flexibility. When we found that rapidly rising student numbers meant our email system of student project submission was imposing an unacceptable load on our staff. Our FLOSS environment meant that developing a rough and ready course management system that provided project acceptance and assessment, with many important automated features, could be developed tested and put into production within three weeks of an inter term break.

4.1 Disadvantages of providing service without resources

There is a down side. If you make a virtue of cheapness in an environment where resources are limited, you will be expected to do things for nothing. Although we accepted hosting requests from other departments and programs who are interested in our extensive features, it turned out unmanageable and eventually we resorted to keeping a low profile on this front.

5 The spread of our influence and difficulties in extending the example

Now we are being asked to propose alternatives for other systems - even to prepare a plan for switching the whole university to open source - that is when our real problems are beginning.

Such a transition requires a concerted effort of university-wide IT department with all their organizational experience and the CS team with their FLOSS experience. However there is obvious -and understandable- fears in the IT team as they will be asked to move away from their core-competences (the proprietary systems). On the other hand lack of larger scale experience prevents the CS team from putting a sensible transition plan on the table to overcome these fears. Thus without a strategic decision to come from management, it does not seem possible to create a critical moment for such transition.

5.1 The danger of being cheap - library example

With the word of mouth going around about open source we were asked for opinion about software systems to be purchased for the University library. Despite our efforts to communicate that deployment of FLOSS systems will require none or little license costs but instead more application labor cost the perception turned out different. We were basically being asked to replace a 150.000 dollar system in 3 months, preferably with no penny put in for additional staffing. Considering the already existent shortage of experienced Linux system developers and administrators in the region this is a real trap! Thus we learned that communicating certain aspects were politically important although they may not seem that way to begin with.

6 Where next?

In today's world campus borders are quite obscure. Although classroom experience is irreplaceable, there is mounting pressure for doing most things through Internet. Thus the next step from the system and service development perspective would be providing students and staff the software environment and

access methods for having computing experience similar to current in-campus arrangement from home, dorm or elsewhere.

Although we are still far from releasing a complete solution many technical aspects for this type of service is already resolved. On the server side, a combination of Kerberos and new generation NFS(NFSv4) provides secure access from off-campus Linux systems. On the client side we have developed enough experience and software to provide CD/DVD-bootable custom Linux systems[1]. Thus ideally a student or staff member can take a CD/DVD as their Linux system with all the software they need, and provided that they have a descent speed Internet connection they can work from any computer. Alternatively they could carry their files on a USB-disk type of removable storage if they do not want to rely on availability of Internet connection.

7 Conclusions

Increased problem solving capacity of our management team was most important result of switching to open source. This included improvement in administration of the system and development of solutions. As open source software packages and their documentation is highly accessible to those involved in management and development of systems, individuals or sub-teams were able to react promptly when problems are reported or requests for new features are received. Unlike proprietary systems which require previous exposure to software documentation, team members were able to fill positions whenever required. After initial period of adaptation to Linux systems, team response time and system error recovery time was considerably improved, although we lack clear logs suitable to present a net measure.

Technically, using Linux systems and FLOSS software enabled us to lower labor and time required for maintenance of our systems, lower hardware costs, and attain a higher level of system integration for better user experience.

However as far as Linux adoption in our wider organization is concerned, we found that after a long period of resistance, management switched to thinking that instant cost savings and painless transitions were possible. Meanwhile the IT department was just itching for us to fail. We now think that a longer transition must be targeted starting with establishing (1)a ground-up experience for the IT team, and (2)a realistic and wider perspective for the management. Thus we may conclude that one should not run into presenting technical aspects and cost-savings perspective without a comprehensive picture of cost of ownership and transition road map.

[1] See https://babbage.cs.bilgi.edu.tr/svnrepos/acarix/trunk/