Re-Thinking Desktop Management

The network is (again) the computer

Tiago Cruz and Paulo Simões
CISUC - DEI
Universidade de Coimbra, Portugal
Email: tjcruz@student.dei.uc.pt, psimoes@dei.uc.pt
http://www.dei.uc.pt

Abstract

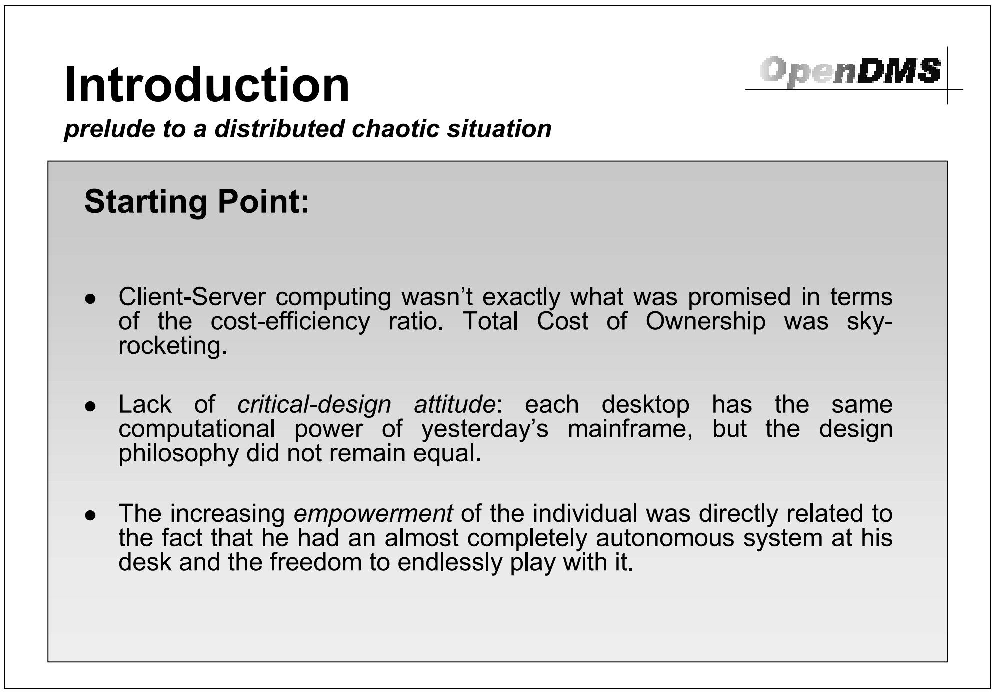
Following the triumph of Moore's Law, power (and, therefore, complexity) was gradually taken off from a central point and distributed over each user's desktop. Now we are facing the price of such a paradigm shift, by ways of an ever-increasing complexity in all matters that respect to network and desktop management. The systems manager faces an almost unbearable pressure in his/her daily work as a result of a simple, indeed unavoidable, physical property of all matter-related things: he/she can't be everywhere.

Although frequently overlooked and considered as "minor work", when compared to network operations and management, the truth is that desktop management is often the most time-consuming task for the typical operations and support team. Our own experience in the management of several small and medium sized institutions (with tenths or hundreds of PCs per location) tells us that desktop management usually occupies more than three-quarters of available human resources, leaving less than one quarter for the management of networks and servers. It is time to start changing this scenario.

The industry has undoubtedly gone to great extents in its effort to provide the means to support desktop management related-tasks. However, the majority of available desktop management platforms suffer from the same problems: feature overkill (a.k.a. *featuritis*), excessive amount of proprietary extensions and too little flexibility.

OpenDMS represents a different response to these problems. In this project we are building an open desktop management framework based on three key ideas. First, the notion that currently available standards and open-source tools already provide a good starting point to create effective management solutions, with no need for proprietary mechanisms. Second, the belief that it is possible to use the network to manage desktops with absent or non-working local file systems or operating systems – in opposition to classical tools where desktop management starts only after the successful load of a working operating system (OS). Such *early management* can be used, for instance, to detect hardware problems, to recover from file system failures or to identify desktop users even before loading the operating system (e.g. in order to implement resource usage policies). Flexibility is the third key idea. Unlike the majority of desktop management tools, which is too much oriented towards a specific configuration (such as the classic Windows Server controlling Windows desktops), the *OpenDMS* framework tries to fit in several desktop paradigms (classic Windows or Unix desktops, Thin Clients, Network Appliances, etc.) and several network configurations.

The first part of this paper addresses the status and pitfalls of desktop management. After describing how the transition from centralized systems to the current distributed PC-based scenario affected desktop management costs, we discuss the currently available desktop management standards and commercial tools. In the second part of the paper we present the key ideas and technical foundations of the *OpenDMS Project*.



Introduction

prelude to a distributed chaotic situation

Starting Point:

- Client-Server computing wasn't exactly what was promised in terms of the cost-efficiency ratio. Total Cost of Ownership was sky-rocketing.
- Lack of *critical-design attitude*: each desktop has the same computational power of yesterday's mainframe, but the design philosophy did not remain equal.
- The increasing *empowerment* of the individual was directly related to the fact that he had an almost completely autonomous system at his desk and the freedom to endlessly play with it.

I. Introduction

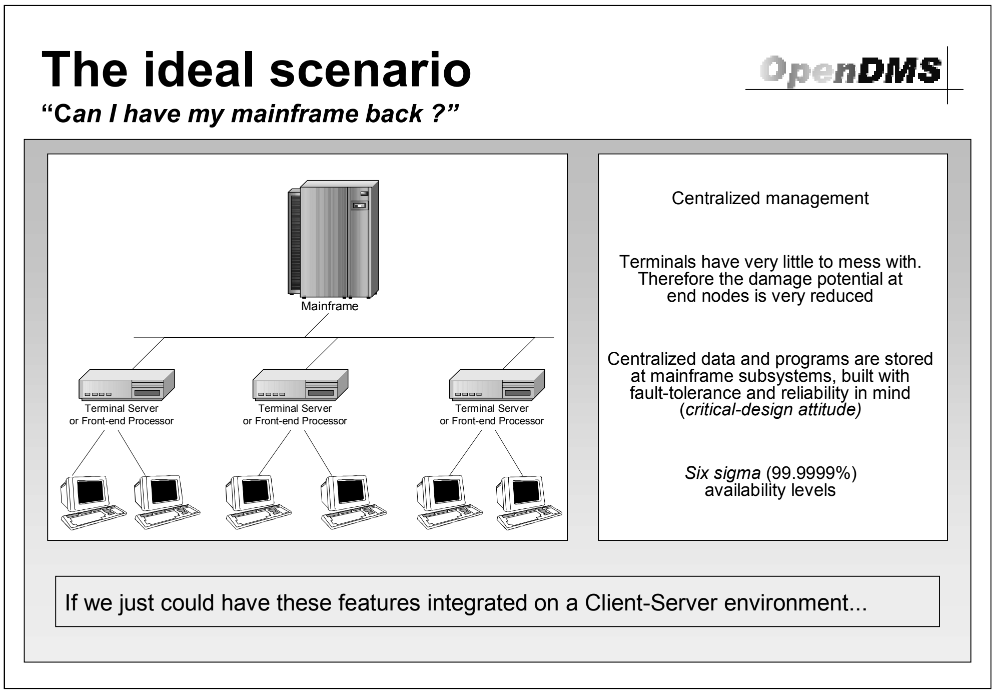
Men are from Mars. Women from Venus. Computers are from Hell.

There was a time in which the dominant computing paradigm was, by concept, based on a centralized management philosophy with a mainframe on top of an hierarchic system with little or no computing power at the end nodes. Hardware was built with reliability in mind simply because when a one thousand-user mainframe crashed it was not possible to just reboot and go on working [1-2]. Customers wanted to know why the system went down and wanted the problem fixed – yesterday.

With the advent of the *client-server* and *workgroup* computing paradigms, every desk got a computer. Decade-old mainframes and dumb terminals were kicked out in favor of a decentralized architecture dominated by PCs, with the promise that the adoption of such ideas would bring *Total Cost of Ownership* (TCO) gains – "*reduce software maintenance costs, increase software portability, boost the performance of existing networks [...] increasing developer's productivity and shortening development lifecycle"* [3]. But the test of time showed that, in many aspects, the client-server architecture failed to keep this promise.

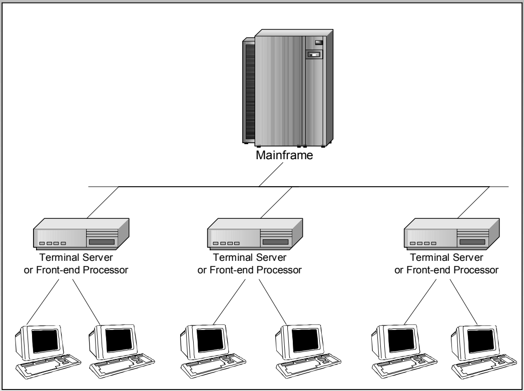
Why? Because PCs ("*the most crash-prone computers ever built*" [1]) are a different kind of animal. Unlike its predecessor computers, PC hardware and software is built with slack reliability requirements, in order to cope with shorter product development cycles and low price targets. Furthermore, being the first truly *personal* computers, PCs did suffer from more user-triggered *creative tweaking* than any other machine before. One way or the other, *core dumps*, *blue screens*, *unexpected application errors* and other varieties of effort-consuming crashes became so frequent that most users just tolerate them as the inevitable price to pay for (apparently) inexpensive computing resources.

While mainframe users tend to care with what's going on, most PC users do not. While a mainframe can chug along for years without crashing, the traditional PC simply can not. If we take a close look at a PC TCO (adding all involved costs: software, hardware and maintenance-related operations) we risk to suffer from an apoplexy. Now, take a deep breath and multiply the individual TCO value by the number of PCs in the average organization. Awful. In some cases even LAN Return Of Investment (ROI) can be seriously compromised [4-5].



The ideal scenario

"Can I have my mainframe back ?"


Mainframe
Terminal Server or Front-end Processor
Terminal Server or Front-end Processor
Terminal Server or Front-end Processor

Centralized management

Terminals have very little to mess with. Therefore the damage potential at end nodes is very reduced

Centralized data and programs are stored at mainframe subsystems, built with fault-tolerance and reliability in mind (*critical-design attitude)*

Six sigma (99.9999%) availability levels

If we just could have these features integrated on a Client-Server environment...

II. The Mainframe Scenario

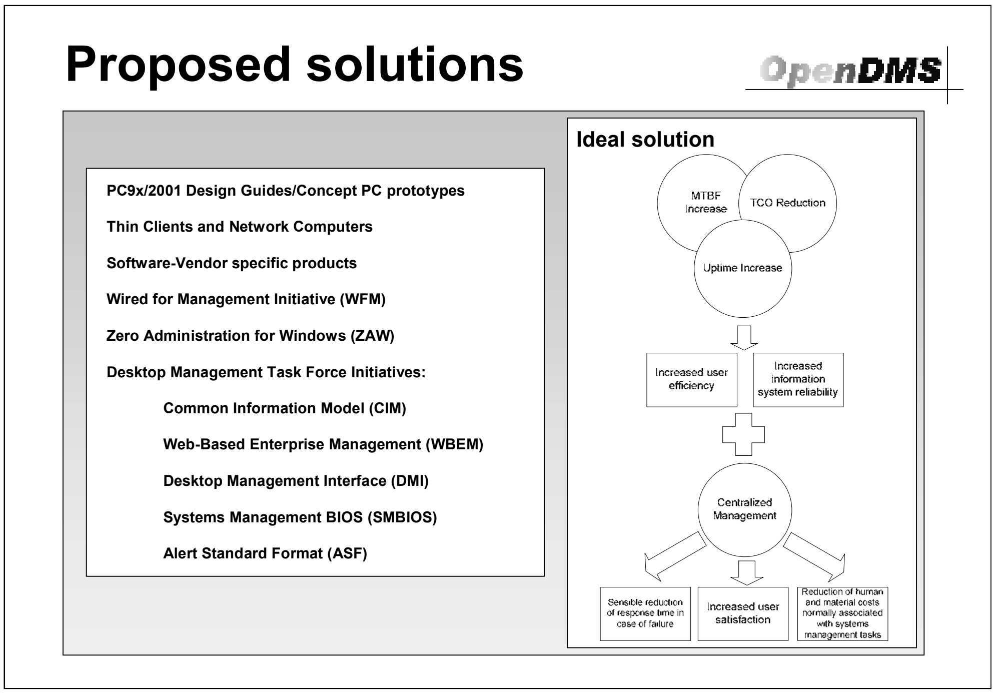
Mainframe: an obsolete device still used by thousands of obsolete companies serving billions of obsolete customers and making huge obsolete profits for their obsolete shareholders. And this year's run twice as fast as last year's

Despite their capacity to work for years without a minute of downtime, the big and expensive mainframes were relegated to be a thing of the past due to maintenance costs and obsolescence, among other reasons. Still, the old *big-iron* systems taught us some lessons that should not be forgotten.

It is undeniable that the mainframe model had serious drawbacks. However there were also some advantages that were forgotten with the pass of time. Rock-solid memory protection, software and hardware designed with critical-attitude in mind, careful administration and an almost complete lack of user freedom – dumb terminals are *stateless* systems without many failure points to deal with – turned mainframes into extremely reliable systems [1-2].

When the *distributed computing paradigm* made its appearance, dumb terminals were replaced by PCs. However, and somehow, it was soon discovered that the expressions "PC" and "user freedom" don't go along together. PCs are *stateful* systems with a considerable number of potential weaknesses completely exposed to the user's "initiative", simply not designed with reliability concerns in mind. From hardware to software, designers and developers resorted to all kinds of tricks to make appearance prevail over performance. *A PC in every desk?* Frightening!

The *so-what* attitude plagued the IT industry as a consequence of "do it yourself" administrators (many PC users insist on being their own *syadmins*), *fatware* (code bloat out of control), weak memory protection and unreliable hardware. And, as features were stacked together, complexity and TCO sky-rocketed [6].



Proposed solutions

OpenDMS

- PC9x/2001 Design Guides/Concept PC prototypes
- Thin Clients and Network Computers
- Software-Vendor specific products
- Wired for Management Initiative (WFM)
- Zero Administration for Windows (ZAW)
- Desktop Management Task Force Initiatives:
  - Common Information Model (CIM)
  - Web-Based Enterprise Management (WBEM)
  - Desktop Management Interface (DMI)
  - Systems Management BIOS (SMBIOS)
  - Alert Standard Format (ASF)

Ideal solution
MTBF Increase
TCO Reduction
Uptime Increase
Increased user efficiency
Increased information system reliability
Centralized Management
Sensible reduction of response time in case of failure
Increased user satisfaction
Reduction of human and material costs normally associated with systems management tasks

III. Proposed solutions

Through the time, several attempts were made to reengineer the PC platform, with ecology, ergonomics and TCO concerns in mind, including the IBM PS/2 Energy Desktop [7-8], the ATX form factor prototype families from *Intel* [9-12] and the ITX family from *VIA Technologies* [13-15]. Power efficiency, ergonomics, flexibility, legacy-free design, aesthetics, small size and use of recyclable materials were the core ideas behind these *concept-PCs*.

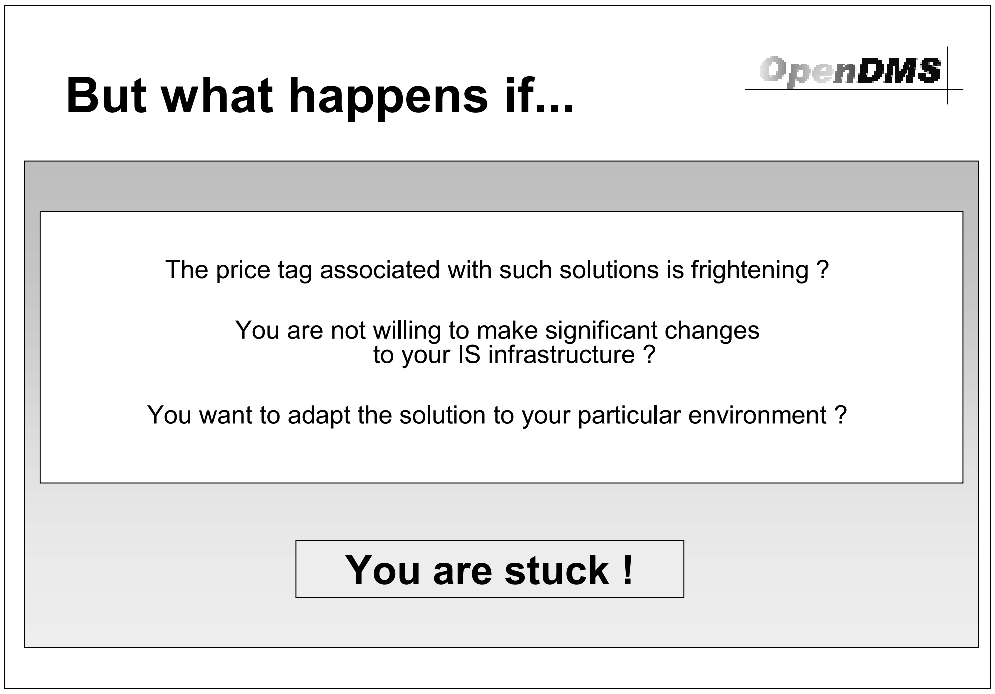
In order to steer and accelerate the PC evolution, Intel and Microsoft joined forces to create the *PC Design Guides* [16]. These guidelines were mainly MS Windows-oriented, but nevertheless their contribution to the PC evolution is undeniable.

Thin Clients and *Network PCs* were presented as the solution to the TCO problem. Reducing user freedom and moving data and programs back to a central system was an appealing solution, but the potential of such technologies never reached critical mass because of technology limitations (*bandwidth,* among others more or less relevant) and the proprietary *open-disguised* approach in which software and hardware were actually closed and very platform-specific. This was the case of *X-Terminals*, Microsoft's *NetPC* and Oracle's *NC.*

A number of worthwhile initiatives were conducted in the context of the *Distributed Management Task Force* (DMTF [17]). From the *Desktop Management Interface* (DMI) to the *Web-Based Enterprise Management*, DMTF provided a whole set of standards that paved the way for initiatives like Microsoft's *Zero Administration for Windows* and Intel's *Wired for Management* [18].

But interoperability is still an issue between products from different vendors. Solutions like Intel's *Landesk* suite and *Microsoft's* SMS [19] are only effective when managing Microsoft-only networks within the boundaries of a very product-specific management approach that complicates interoperability. Furthermore, partial implementations of DMTF standards are very common (e.g. some motherboards have DMI/SMBIOS [20] compliant firmware that does fill the system information table, but with incorrect or no data at all).

The ideal solution should answer to a specific set of needs following a bottom-up approach, addressing IS-wide TCO, MTBF and reliability sensitive aspects through clear and well-defined asset acquisition policy guidelines and follow-up documentation with user efficiency and information system reliability concerns. Recentralizing control as much as possible (within reasonable limits) through the use of adequate tools and technologies. Although avoiding to turn PCs back into dumb terminals by imposing excessive limitations, an important amount of centralized control must be restored in order to bring order to the IS infrastructure.



But what happens if...

OpenDMS

The price tag associated with such solutions is frightening ?

You are not willing to make significant changes to your IS infrastructure ?

You want to adapt the solution to your particular environment ?

You are stuck !

IV. But what happens if...

"Try not. Do or do not. There is no try." - Master Yoda (Starwars)

Many systems administrators took the time and the effort to convince the upper-management layer to adopt one of the well-known commercial desktop management products. First question they faced: "How much will it cost?" – "too much" is the right answer in many cases. Take, for example, Microsoft's *SMS Server* in a network with 90 client PCs. You'll need to acquire the following items:

1 x Microsoft SQL Server License

1 x SMS 2.0 w/ 10 Client Access License

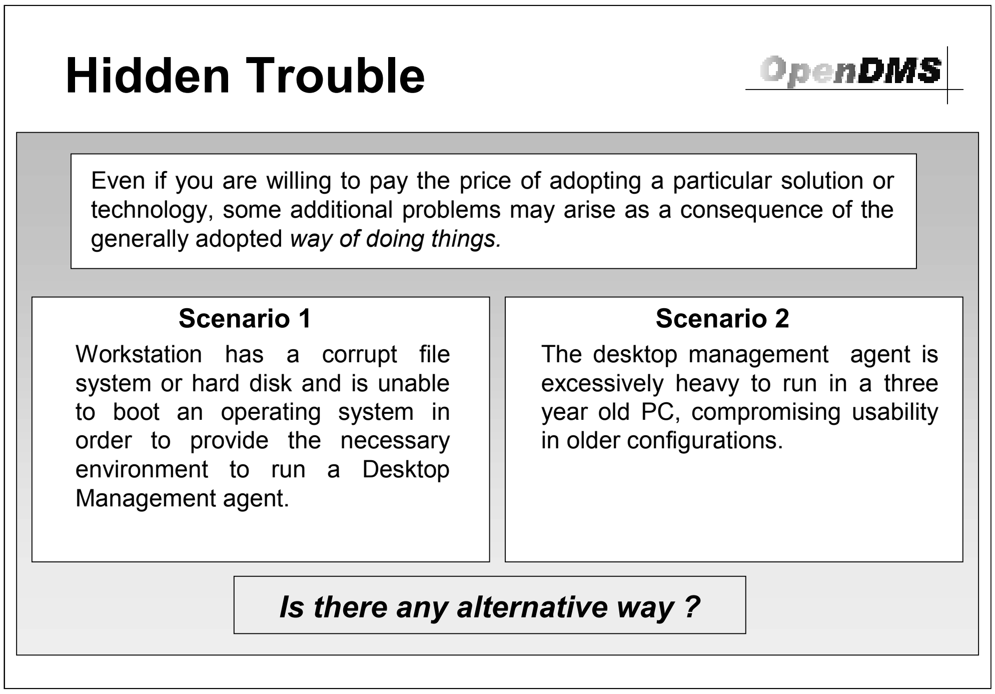
4 x 20 Client Access License Pack

1 x Windows 2000 Server Access License

Now take in consideration that *SMS Server* is oriented just towards asset-management, basic troubleshooting and software distribution related-tasks. There is no remote PC health monitoring, for example. Therefore, this is a rather limited solution. Although these limitations can be solved with third-party tools, this often means duplicating existing functionality and rising cost even more. And if we take in consideration the associated deployment costs (server/client downtime and work time), things only tend to get worse.

Thin clients had their own problems too, as soon found out by those who swallowed the pill and bet on *NCs* or *NetPCs.* Thin clients, usually based on proprietary hardware and software, were more expensive and less upgradeable than stock PCs and imposed significant changes in the normal client-server environment. Thin clients were also affected by bandwidth problems, given their inefficient design and the inherent limitations of LAN technology at the time (most thin client proposals date from 1997, a time when even switched *fast-ethernet* to the desktop could become relatively expensive). Their inefficient design also meant heavy investments in additional server horsepower. Besides, a large amount of applications could not fit this new paradigm without extensive porting.

Assume you have the latest, greatest distributed management suite and you don't need all those bells and whistles that came with it, but only a small subset. Or, even worse, you have a *Microsoft SMS*-dependent infrastructure in place and you need to add PC-based non-Windows workstations. You know *SMS* does not support *windows-less* PCs but you are willing to spend some time adapting the solution to your needs. What now? Code is closed, communication formats are unknown or depend on the OS and you are completely stuck.



Hidden Trouble

OpenDMS

Even if you are willing to pay the price of adopting a particular solution or technology, some additional problems may arise as a consequence of the generally adopted *way of doing things.*

Scenario 1

Workstation has a corrupt file system or hard disk and is unable to boot an operating system in order to provide the necessary environment to run a Desktop Management agent.

Scenario 2

The desktop management agent is excessively heavy to run in a three year old PC, compromising usability in older configurations.

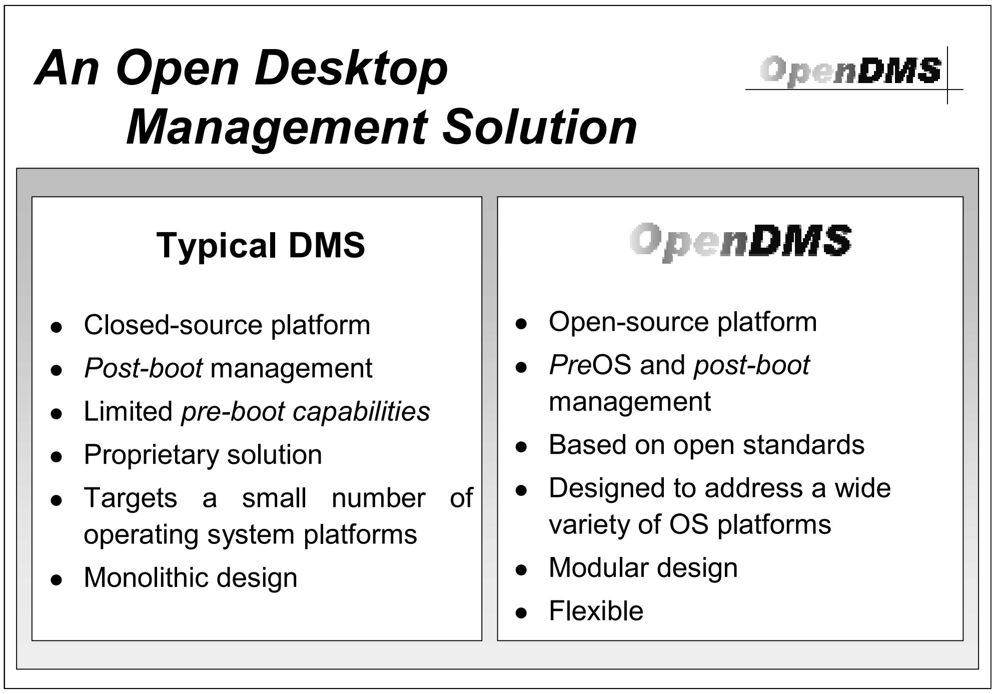
Is there any alternative way ?

V. Hidden Trouble

Even considering you do not feel restricted by the limitations of the industry-proposed solutions, there are some inevitable problems that, sooner or later, will haunt you. Those hidden limitations are consequence of design-flaws in the management suites themselves. Two examples of such common situations are:

• somewhere in the enterprise a workstation has its file system seriously damaged. The system administrator knows what happened because a user made a phone call requesting helpdesk intervention. It would be nice if the system administrator could remotely lock the PC, preventing further damage until someone qualified arrives, but unfortunately the remote management suite capabilities were not designed to work in OS-absent environments. It is possible to have some kind of *pre-boot* capabilities, but they are only oriented to local OS-distribution. So, the solution is to send someone to fix the problem locally.

• the system administrator has the greatest, feature-packed distributed management suite at his disposal. Client Desktop Management Agents even have local web-server capabilities with lots of blinking lights and animated features. But the typical PC at the organization can not support the additional load carried by the management agent without an upgrade. It shouldn't be like that.



An Open Desktop Management Solution

Typical DMS	OpenDMS
• Closed-source platform • *Post-boot* management • Limited *pre-boot capabilities* • Proprietary solution • Targets a small number of operating system platforms • Monolithic design	• Open-source platform • *Pre*OS and *post-boot* management • Based on open standards • Designed to address a wide variety of OS platforms • Modular design • Flexible

VI. An Open Desktop Management Solution

Typical desktop management solutions are normally designed to work in the presence of a local working OS, and offered *pre-boot* capabilities are often reduced to OS installation or image distribution. With this monolithic approach the managed desktop needs a full-blown working OS and, on top of that, a desktop management service with a whole set of features, including a lot of unnecessary ones. Nevertheless, despite this overkill setup, the systems manager will not be able to add extra functionality because the source code is closed and OS-dependent.

As an alternative to the available solutions, the *OpenDMS* framework supports *OS-absent* operations. Instead of relying on the presence of an operating system to start managing the desktop, the *OpenDMS* framework pushes as many management functionality as possible to the moment before the PC loads its OS. At this moment the requirements in each managed node are minimal. All is needed is a reduced set of healthy hardware components (power supply, motherboard, memory, processor, network adapter and, eventually, console devices to perform locally/user initiated helpdesk-related operations), a network connection (to communicate with the *OpenDMS* server and other network services) and remote boot firmware extensions (a common feature in current Network Interface Cards).

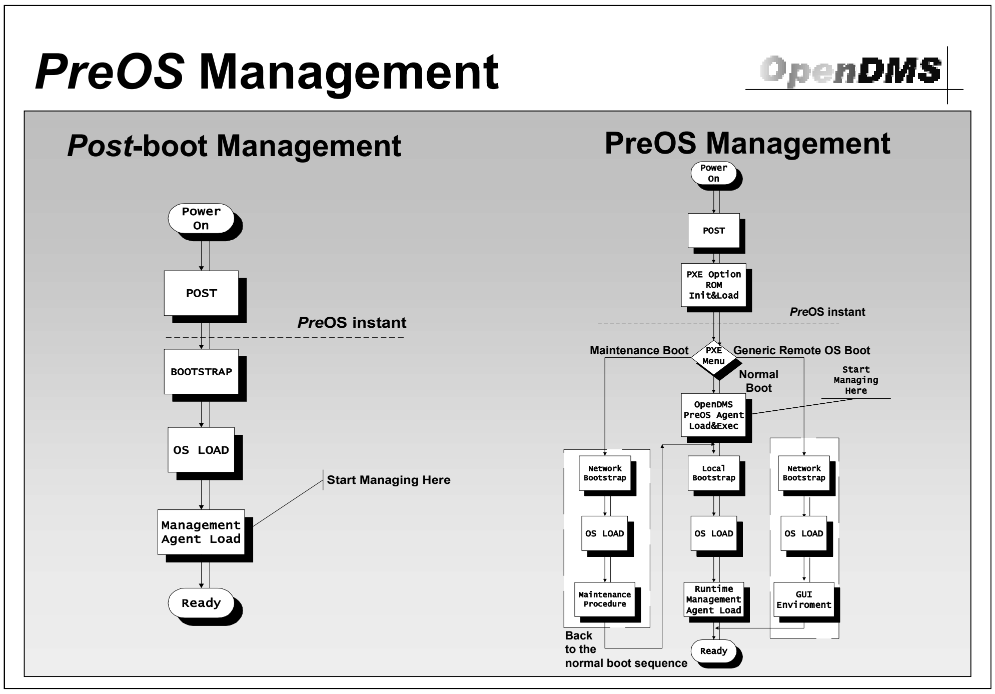
There are good reasons to make PCs remotely manageable right after *Power On Self-Test* (POST) procedures.

The first reason is the convenience of minimizing technical staff displacements to assist in helpdesk-related tasks, in order to concentrate in preventive maintenance operations. Even if the local operating system is not working, the user may call for interactive helpdesk assistance from the desktop console. If there is a problem with the local file system the PC may be locked to prevent further damage, and in some cases it is even possible to remotely repair or replace the local OS image.

The second reason is increased control. *Pre-boot* management can be used to check the health of hardware, local file systems and even local operating systems. Combined with early user authentication, this provides a means to implement flexible resource usage policies. According to the user and the circumstances, different operating systems, different configurations or different computational resources may be selected *a priori*, enforcing management policies stored in a single location (the *OpenDMS* server).

Another reason is the fact that in this way it is possible to reduce the footprint of classical (*OS-present*) management services. *Pre-boot* management is obviously limited, and therefore *online management services*, in the way they are provided by most desktop management tools, are still necessary. However, since many tasks are now performed at boot time, the *online service* overhead on the client system is reduced.

The *OpenDMS* framework mixes *pre-boot* and online management in order to bring more control back into the hands of operations and support teams, putting the network infrastructure in the center of the distributed management effort and recentralizing management capabilities through a modular, open-source and platform neutral approach based in open standards.



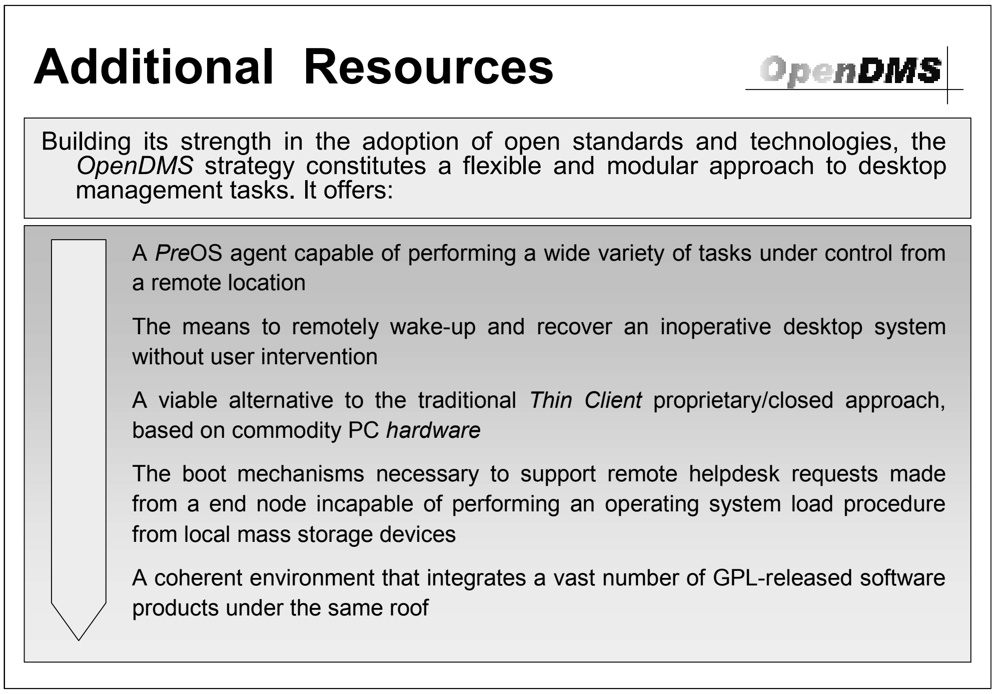
VII. *PreOS* Management With *OpenDMS*

The *OpenDMS* approach for *pre-boot* management relies in a simple but important concept: the so-called *PreOS* instant of the PC initialization sequence. Making a remotely-controlled detour in this precise instant, in order to execute a special *PreOS agent* or to boot an OS over the network, it is possible to manage a desktop PC without a working local OS – a network connection and minimally functional hardware is all we need. Luckily, the standard PC architecture was designed to support this kind of special-purpose firmware extensions through the use of *option-ROMs*, which are normally embedded in special purpose hardware such as *Network Boot ROMs.* These extensions provide the means to locally boot an OS previously downloaded from remote servers or, with a little tweaking, specific-purpose software such as the *PreOS agent*.

OS-absent operations are supported by a specific type of *Network Boot ROM* firmware extension, commonly known as *Preboot eXecution Enviroment (PXE) Boot ROM* [21]. Created in the context of the *Intel Boot Initiative* [22], four years ago, the PXE Boot ROM is now widely recognized as the standard for remote boot ROMs. It is included in the vast majority of Network Interface Cards, including *Lan On Motherboard* [23] designs in which the firmware extension is embedded as a module in the same chip as the manufacturer's BIOS. The forthcoming specification of the Intel *Extensible Firmware Interface* [24] (a replacement for the currently available BIOS architecture for IA-32 systems, already available for IA-64 systems) will also support PXE.

PXE-compliant Boot ROMs provide means to control the boot process in order to download and execute either a full-blown *OS* or just a small *PreOS management agent.* In an *OpenDMS-managed* desktop PC the normal boot sequence from local mass storage is preceded by a selection stage at the *PreOS* instant. At this point it is possible to boot a remote OS (in order to initiate maintenance tasks) or to proceed with the normal boot sequence. In this case a *PreOS agent* is downloaded and executed before the local OS-load from mass storage. The *PreOS agent* may then check hardware inventory data (using DMI/SMBIOS or direct access methods), lock the boot process (avoiding further progress and shutting down the system, if needed), ask for authentication information (that may be used, for instance, to decide which OS and which environment will be loaded), attempt to repair local file systems, etc. These operations, including the selection stage, are either locally triggered by the user (if desirable) or remotely controlled by the *OpenDMS* server.

The additional "*Generic Remote OS Boot*" procedure presented in the figure has a special purpose: it shows that it is possible, for instance, to have a remotely loaded full-blown OS with graphic environment available to the user (e.g. *Linux + X/Windows*). This makes possible to have desktop PCs with *Thin-Client personality* capable of booting an alternate environment (without requiring local installation) that can be used, for instance, to make a helpdesk service request using a browser in a PC incapable of booting from a local hard disk (because of physical damage or file system corruption). But there is more to say about this capability, as we will discuss later on.



Additional Resources

Building its strength in the adoption of open standards and technologies, the *OpenDMS* strategy constitutes a flexible and modular approach to desktop management tasks. It offers:

A *Pre*OS agent capable of performing a wide variety of tasks under control from a remote location

The means to remotely wake-up and recover an inoperative desktop system without user intervention

A viable alternative to the traditional *Thin Client* proprietary/closed approach, based on commodity PC *hardware*

The boot mechanisms necessary to support remote helpdesk requests made from a end node incapable of performing an operating system load procedure from local mass storage devices

A coherent environment that integrates a vast number of GPL-released software products under the same roof

VIII. Additional Resources

Pre-boot management capabilities are just one of the components of the whole *OpenDMS* framework.

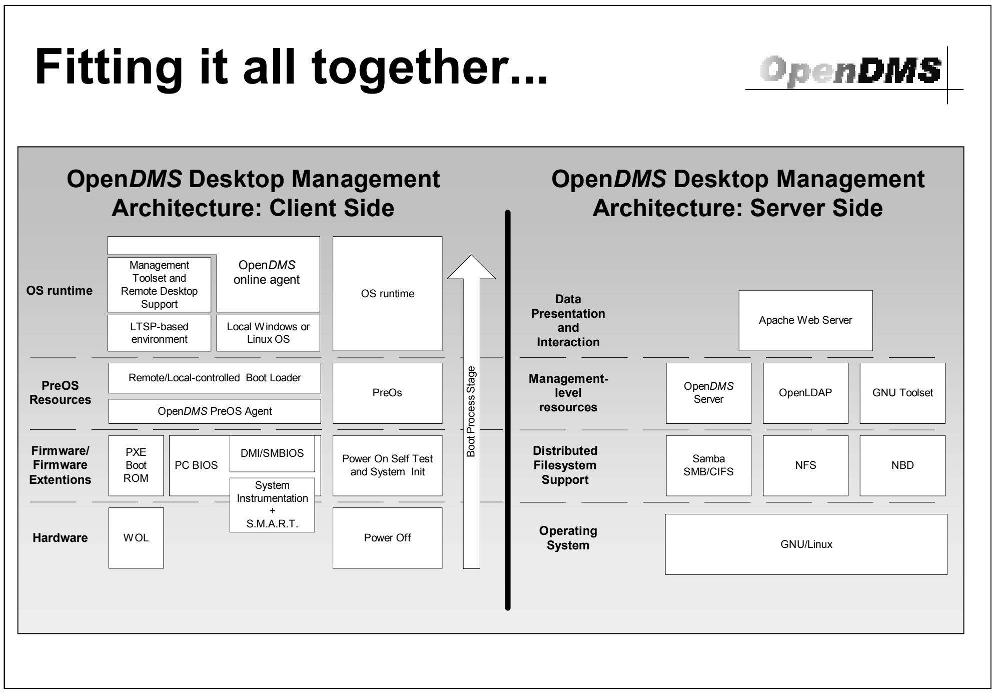
The *OpenDMS* also includes an online management service which is similar to those provided by commercial products but not so feature rich – both because some functionality is already provided by the *PreOS agent* and because it was decided not to implement many architecture specific features, in order to keep a lightweight and platform-neutral solution. Currently there are online management services for Windows and for Linux, providing basic system monitoring, helpdesk assistance and a remote desktop service (based on the VNC) for more complex management tasks.

The OpenDMS server controls both *pre-boot* management and *online* management, according to policies stored using *OpenLDAP* directory services [25]. It also supports off-hours desktop maintenance (backup, virus scanning, etc.) trough *Wake-On-LAN* (WOL) [26-27].

The support of alternatives to the classical full-blown Windows desktop lead to the specification of a reference thin client platform specification. This thin client is built from commodity PC hardware and is able to connect to several platforms, including character-based protocols, Microsoft Remote Desktop-based systems, X/Protocol servers, VNC (*Virtual Network Computing* [28]) and Citrix Metaframe [29] (client not available under GPL). It is also able to access existing local multimedia and local or remote storage resources (using, for instance, NFS or SMB/CIFS). The list of potential applications includes desktop hybrid *NCs-PCs*, multimedia kiosks and network appliances.

The *OpenDMS* project includes the thin client reference specification, software modules to build desktops according to this model, OpenDMS Server support for deployment and management of thin clients, and also some tools and guidelines to connect thin clients to network resources.

It should be stressed that the *OpenDMS* framework is built upon open standards and, whenever possible, makes use of already available open-sourced products such as, for instance, OpenLDAP, Samba and VNC. With the right capabilities enabled and at our disposal, there was an excellent foundation to start the groundwork for the *OpenDMS* framework. However, a question remains: “How it all glues together ?”- that is something we will see next.


Fitting it all together...
OpenDMS
OpenDMS Desktop Management Architecture: Client Side
OpenDMS Desktop Management Architecture: Server Side
OS runtime
Management Toolset and Remote Desktop Support
OpenDMS online agent
LTSP-based environment
Local Windows or Linux OS
OS runtime
PreOS Resources
Remote/Local-controlled Boot Loader
OpenDMS PreOS Agent
PreOs
Firmware/ Firmware Extentions
PXE Boot ROM
PC BIOS
DMI/SMBIOS
System Instrumentation + S.M.A.R.T.
Power On Self Test and System Init
Hardware
WOL
Power Off
Boot Process Stage
Data Presentation and Interaction
Apache Web Server
Management-level resources
OpenDMS Server
OpenLDAP
GNU Toolset
Distributed Filesystem Support
Samba SMB/CIFS
NFS
NBD
Operating System
GNU/Linux

IX. Fitting it all together…

The *OpenDMS* approach distinguishes itself by the use of a whole set of technologies and *GPL-available* resources glued together in order to achieve a defined set of goals. Leveraging the potential of specific purpose components and technologies, sometimes in a different context from the one they were originally intended for, the *OpenDMS* framework tries to integrate them under a single management platform, designed with heterogeneous environments in mind and capable of interacting with a wide range of OS environments.

GNU/Linux was chosen to build the management server for its robustness and because it comes bundled with a tried-and-tested set of useful services. The same OS also provided the basic environment for the *thin-client* platform, using the LTSP framework (*Linux Terminal Server Project* [30]).

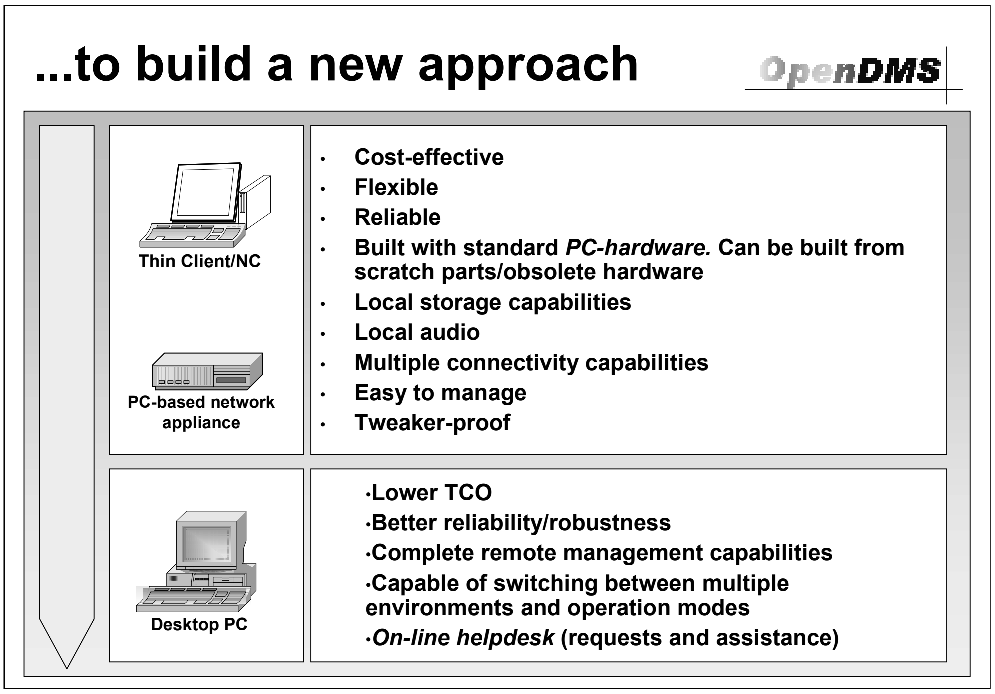
The *Apache Web Server* was installed on the central management server in order to provide a Web-based management console to the systems manager. *OpenLDAP* was used to build the directory services where management information is stored and organized.

Several distributed file systems are supported, including Samba-based SMB/CIFS (*Server Message Block/Common Internet File System* [31]) for accessing file and print services located on MS-Windows systems, NFS (*Network File System*) to access remote file system resources in Unix-like systems, and NBD (*Network Block Device* [32]) for seamless access to mass storage devices located in thin clients.

Local file systems are limited to Linux-supported file systems, since the management boot is based on a remotely downloaded GNU/Linux OS instance. Nevertheless, this is more that enough for our purposes.

There is also support for a wide range of remote console protocols: remote text-mode console protocols such as *telnet* and *ssh;* VNC, which is available in a wide range of platforms; X-windows; and RDP (*Remote Desktop Protocol* [33]), used to communicate with Windows Terminal Server (NT and 2000 systems) and XP systems (using Remote Assistance Capability).

All these features, combined with the *preboot* and *online* management components (Wake on LAN, PXE, DMI/SMBIOS instrumentation, PreOS agent and on-line agent) ensure that the *OpenDMS* framework is a flexible and modular solution for today's desktop management requirements.



...to build a new approach

OpenDMS

Thin Client/NC PC-based network appliance	· Cost-effective · Flexible · Reliable · Built with standard *PC-hardware.* Can be built from scratch parts/obsolete hardware · Local storage capabilities · Local audio · Multiple connectivity capabilities · Easy to manage · Tweaker-proof
Desktop PC	·Lower TCO ·Better reliability/robustness ·Complete remote management capabilities ·Capable of switching between multiple environments and operation modes ·*On-line helpdesk* (requests and assistance)

X. ...to build a new approach

The *OpenDMS* framework provides a whole range of possibilities to the systems manager. It becomes possible to envision a scenario where the classic desktop PC stops being the only choice available to the user.

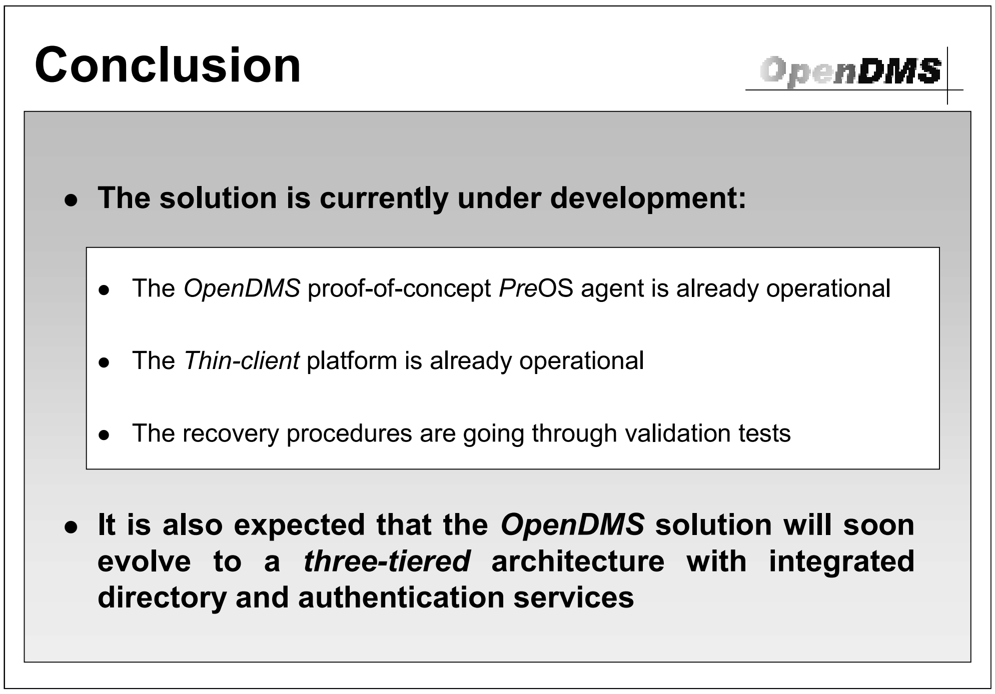
The *thin client* approach seems to have some relevance if we take into account the fact that it shares the same benefits obtained from dumb terminals while remaining user-friendly. If we remove some obstacles (like proprietary *hardware/software* components) while offering a cost-effective option with decent bandwidth, the *thin client* solution could have the potential to become a viable alternative in real-world scenarios. That is what we tried to do: using PXE Boot ROMs associated with a specially modified mini-Linux distribution (based on LTSP) it is possible to build a PC-based *thin-client* with local capabilities such as audio and storage (eventually limited to removable and/or fixed low-capacity devices). Because they have very little mechanical moving parts (possibly just a power supply and a CPU cooler fan in minimalist configurations, but even those can be eliminated through the use of quality hardware) *OpenDMS thin clients* are potentially reliable, *stateless* systems with advanced graphic and multimedia capabilities. Furthermore, these *thin-clients* may be connected to a wide range of systems, either using text mode consoles (telnet, ssh, terminal emulation modes) or graphic remote-desktop protocols (X/Protocol, RDP/Windows Terminal Server Family, Citrix, VNC). It also becomes possible to build network appliances, like print servers, resorting to the same techniques used to build regular *thin clients,* but in a scaled-down manner.

Simultaneously, regular desktop PC systems will benefit from better management through lower TCO and increased reliability, thanks to features like a better helpdesk service and preventive maintenance. *On-line helpdesk* procedures (request and assistance), together with remote management capabilities, help to reduce costs while increasing operations support *staff* effectiveness – remember that there is no need to displace someone to solve a problem when we can power up a PC remotely. The same resources used to enable *OpenDMS thin clients* can be used to allow a full-blown PC to behave like a *thin client* when needed (e.g. access to a Windows Terminal Server system).

XI. Conclusion and Future Work

In this paper we presented the main features of the *OpenDMS* framework. This approach is not a solution for every desktop management problem, but it does provide a different way of doing things in the field of desktop management.

OpenDMS is a large and ambitious framework. For this reason, despite the large number of useful modules already developed and publicly available, a lot of work still remains to be done. We are currently adding several new capabilities to the *PreOS* agent, namely SMART (Self-Monitoring, Analysis and Reporting Technology [33]) hard disk monitoring support, loadable functional extension module support and secure remote authentication. In the area of authentication we are also working in the integration of external authentication schemes (using MIT Kerberos [34]) and LDAP directory services [35].



Conclusion

OpenDMS

- **The solution is currently under development:**
  - The *OpenDMS* proof-of-concept *Pre*OS agent is already operational
  - The *Thin-client* platform is already operational
  - The recovery procedures are going through validation tests
- **It is also expected that the *OpenDMS* solution will soon evolve to a *three-tiered* architecture with integrated directory and authentication services**

References

[1] *Tom Halfhill, "Here's why today's PC's are the most crash-prone computers ever built – and how you can make yours more reliable*", Byte Magazine, April 1998

[2] Mark Schlack, "*Speed thrills, but crashes kill. Time to focus on quality code and never-say-die computing*", Byte Magazine, May 1998

[3] Alex Berson, "*Client/Server Architecture*", McGraw-Hill Computer Science Series, pp. 11-12

[4] Gartner Consulting, "*TCO Analyst: A White Paper on GartnerGroup's Next Generation Total Cost of Ownership Methodology*", 1997, pp. 19-20

[5] Gerry Blackwell, "*Assessing Total Cost Of Ownership*", 802.11 planet, http://www.80211planet.com

[6] Gartner Consulting, "*TCO Analyst: A White Paper on GartnerGroup's Next Generation Total Cost of Ownership Methodology*", 1997, pp. 10-12

[7] IBM Corporation, "*IBM PS/2 E product information*", IBM DC 00210-00, 1993

[8] L. Clause, "*Energy Desktop: IBM réinvente l'ordinateur de bureau*", Science&Vie Micro, March 1993, pp. 82-85

[9] N. Sumrall, "*High Concept Comes to the PC: Form, Function and Fashion*", Report from Intel Developer Forum, Fall'99, Intel Developer Update Newsletter

[10] Intel Corporation, "*MicroATX Motherboard Interface Specification*", 1997

[11] Intel Corporation, "*FlexATX Addendum Version 1.0 to the microATX Specification*", 1999

[12] Intel Corporation ,"*Ease of Use Initiative – Concept PC*", 2002

[13] VIA Technologies, "ITX Mainboard Specification White Paper", 2001

[14] VIA Technologies, "Mini-ITX Mainboard Specification White Paper", 2002

[15] VIA Technologies, "Information PC reference design", 2001

[16] Intel Corp, Microsoft Corp, "*PC Design Guide Website*", http://www.pcdesguide.org/

[17] DMTF, Distributed Management Task Force, http://www.dmtf.org

[18] L. Weller, "*Reducing TCO with Intel WFM and Microsoft ZAW initiatives*", Intel Developer Update Magazine Issue 17, February 1999

[19] E. Willansky, "*Microsoft's SMS helps bring the anarchy of multiple-vendor networks under centralized control*", Byte Magazine, May 1995

[20] DMTF, "System Management BIOS Reference Specification v2.3.2", 2001

[21] Intel Corp, "Preboot Execution Environment (PXE) specification version 2.0", 2000

[22] Intel Corp, "Intel Boot Initiative program" ,1998

[23] Intel Corp, "*What is LOM ?*", 2000

[24] Henry, Mike, "PXE Manageability Technology for EFI", Intel Developer Update Magazine, October 2000

[25] OpenLDAP, http://www.openldap.org

[26] AMD Corp., "Magic Packet Technical White Paper", 1995

[27] Microsoft Corp., "*Network Device Class Power Management Reference Specification* ", 2000

[28] AT&T Cambridge Labs, "*Virtual Network Computing*", http://www.uk.research.att.com/vnc/

[29] Citrix Metaframe Homepage, http://www.citrix.com/

[30] LTSP, "*Linux Terminal Server Project*", http://www.ltsp.org

[31] Samba Project, http://www.samba.org

[32] NBD Project, http://nbd.sourceforge.net

[33] Chapman, Mark, "*Rdesktop project*", http://rdesktop.sourceforge.net

[34] Maxtor Corp., "*SMART advanced drive diagnostics white paper*", 2001

[35] MIT,"*Kerberos: The Network Authentication Protocol*", http://web.mit.edu/kerberos/www/