FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's thesis in computer science

Application virtualization on Windows 7 with Microsoft App-V

Manuel Söhner

FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's thesis in computer science

Application virtualization on Windows 7 with Microsoft App-V

Applikationsvirtualisierung unter Windows 7 mit Microsoft App-V

Author:	Manuel Söhner
Supervisor:	Prof. Dr. Uwe Baumgarten
Submission date:	November 15, 2011

I assure the single handed composition of this bachelor's thesis only supported by declared resources.

Munich, November 15, 2011 Manuel Söhner

Acknowledgments

There are some people that have contributed to the thesis so I would like to take this occasion to thank them.

First of all, I would like to thank oLLi for offering me the chance to work with the school network, coaching me during the internships and teaching me the SMC's mode of operation.

This thesis would perhaps not have come about if Gabriel had not been so kind to organize a meeting with the client and server team in order to find a convenient topic for all involved parties. So, I would like to thank him, Michael T., Michael K. and all the other contributors of ZIB for giving me the opportunity to research the application virtualization topic and for allowing me to work in the Windows 7 project.

Furthermore, I like to express my gratitude to Christof for the great collaboration and to Robert for the introduction into the AE-Manager and showing me how to deploy the applications. Additionally, I would like to thank the whole development team (including Norbert, Stefan, Carsten, Max and Dirk) for the good time and support.

Abstract

Virtualization has established itself as a helpful concept in data centers as well as on desktop clients. While hardware and operating system virtualization is very common on servers and for test environments, the virtualization of applications on an end-user's workstation is not that widely used yet. Certainly, many company networks consist of various programs that cause conflicts and require a high amount of management effort in solving the problems that occur at installation time and those that appear much later without an obvious correlation. Here, the vendors of application virtualization products are eager for providing a way out.

This thesis will discuss the benefits of application virtualization and which advantages arise for customers that apply such products in order to solve conflicts between programs.

The first part of the thesis will give an introduction to the topic by depicting how application virtualization works and how it differs from other virtualization concepts. Thus, the second chapter will describe how a virtualization layer intercepts function calls in order to reach the isolation and abstraction from the operating system.

As the thesis includes an external cooperation with the bureau of information technology at Munich's department of education, the second part will cover the practical chapters. Here, App-V, the application virtualization product of Microsoft will be explained in more detail as well as its integration into the heterogeneous network of Munich's school network. There, it will be a part of the Windows 7 migration project and is intended be a solution for conflicts that arise by installing incompatible applications on the same machine. Thus, the thesis also observes the pitfalls that may arise when virtualizing programs and it will show which applications cannot be transformed into isolated packages.

The last part will introduce application virtualization products from VMware, Citrix and Symantec by outlining their differences to Microsoft App-V.

Contents

Acknowledgements **vii**

Abstract **ix**

1. Introduction **1**

1.1. Motivation 1

1.2. Types of virtualization 2

1.2.1. Hardware virtualization 2

1.2.2. Operating system-level virtualization 3

1.2.3. Application virtualization 3

1.3. Benefits of application virtualization 5

1.4. Drawbacks of application virtualization 6

1.5. History 7

1.5.1. The time before the Windows Registry 7

1.5.2. After the release of the Windows Registry 8

1.6. Structure of this thesis 9

2. Background **11**

2.1. Architecture of Windows 11

2.2. Placing the virtualization engine 12

2.2.1. Interception in user mode 12

2.2.2. Interception in kernel mode 13

2.3. Functionality of a virtualization engine 14

2.4. Windows API hooking 16

2.4.1. Basic mechanism for intercepting function calls 16

2.4.2. Microsoft Detours 17

2.5. Intercepting requests to the file system and registry 18

3. Microsoft Application Virtualization (App-V) **19**

3.1. History 19

3.2. Functionality of App-V 20

3.3. Constraints of App-V 23

3.4. Components 25

3.4.1. Sequencer 25

3.4.2. Management Server 28

3.4.3. Web Service, Data Store, and Management Console 29

3.4.4. Streaming Server 30

3.4.5. Client 30

3.4.6. App-V for Remote Desktop Services 32
3.5. Architecture 33
3.5.1. Full infrastructure 33
3.5.2. Stand-alone mode 34
3.6. Interaction between isolated applications 35
3.7. Server Application Virtualization 37

4. Integrating App-V into a heterogeneous network **39**
4.1. Infrastructure of Munich's school network 39
4.2. Reverse engineering the infrastructure of Microsoft App-V 40
4.2.1. Streaming and authorization 40
4.2.2. Publishing and authentication 41
4.2.3. Reporting 42
4.3. App-V as part of the Windows 7 migration project 42
4.3.1. App-V Client 44
4.3.2. Mobile users 44
4.4. Software 45
4.4.1. Adobe Reader 45
4.4.2. Adobe Creative Suite and Adobe Acrobat 47
4.4.3. Hardcopy 48
4.4.4. Microsoft Office 48
4.4.5. Mozilla Firefox and Thunderbird 49
4.4.6. Investigation of problems inside the virtual environment 51
4.4.7. Documentation of proceedings and troubleshooting 51
4.5. Categorization of applications 52
4.5.1. Not virtualizable 52
4.5.2. Virtualizable, with reservations 53
4.5.3. Virtualizable 54

5. App-V compared to other products **55**
5.1. VMware ThinApp 55
5.2. Citrix Application Streaming 56
5.3. Symantec Workspace Virtualization 57

6. Conclusion **59**

Appendix **63**

A. App-V documentation for ZIB **63**

Bibliography **87**

List of figures

1.1. Native hypervisor . . . 3
1.2. Hosted hypervisor . . . 3
1.3. Operating system virtualization . . . 4
1.4. Application virtualization . . . 4

2.1. Simplified architecture of Windows . . . 12
2.2. Interception on the Native API level . . . 13
2.3. Interception in kernel mode . . . 14
2.4. Redirection of requests by the virtualization layer . . . 15
2.5. Situation before and after the insertion of the detour . . . 17
2.6. Positioning of the file system and registry filter . . . 18

3.1. Virtualization layer of App-V . . . 20
3.2. Direct file requests to the virtual drive and redirected ones to the system drive 22
3.3. Monitoring an installation . . . 25
3.4. Launching process of an application . . . 31
3.5. Infrastructure of App-V . . . 33
3.6. Two independent Dynamic Suite Compositions . . . 36

4.1. App-V integrated into the school network . . . 43

List of tables

4.1. List of software that is part of the Windows 7 pilot experiment 46

Listings

3.1. Directory structure of the virtual drive 22
3.2. Command for launching Microsoft Office Word 2010 31

4.1. Command for launching the registry editor inside the virtual environment . 51

1. Introduction

Installing applications on the personal computer at home normally works without noticing any problems. Sometimes the operating system can be corrupted after an installation and fail to start, which is mostly caused by device drivers. Certainly, it is more likely that programs conflict with each other, rather than damaging the whole system.

However, in networks with many desktop clients and different applications the likelihood for conflicts between programs is much higher than on a stand-alone computer that is used at home. Oftentimes, companies and government agencies apply special software to fulfill the daily work and thus have a high interest in running the programs on a machine in parallel as much as possible. This is not always feasible because the software may have unequal requirements, use different versions of libraries or modify the system in a way that makes it incompatible to other applications. Also, running different versions of the same product is mostly a problem but is sometimes required by the customers.

This is where application virtualization comes into play as it aims to eliminate conflicts between applications by isolating them from the operating system and other programs. Application virtualization products like Microsoft App-V, VMware ThinApp or Citrix XenApp promise to help businesses by solving most of the problems that arise with the installation of programs [21, 56, 52].

This thesis will cover the application virtualization topic by explaining its functionality and discussing Microsoft's product with its components in more detail. As the topic was offered by the bureau of information technology, which provides services for Munich's department of education, the thesis also includes the implementation of Microsoft App-V into their network to analyze whether App-V can be used to solve the issue of conflicting applications. Thereto, I researched on how to integrate the product into a heterogeneous network and investigated which applications are candidates for being virtualized and which are not.

1.1. Motivation

The history of virtualization dates back to the 1960's and there is still ongoing research these days [82]. Today, the analysis of this topic is mainly focused on security and the opportunities of virtualization for cloud computing. Thus, since a few years, many administrators use virtualization in data centers to run several isolated systems on one machine. This not only reduces the costs of operation because of the saved hardware, electricity, and easier management. It also saves administration time and makes the systems more secure and portable.

Virtualization also enables server administrators to separate different services (like a database and the web server) from each other by installing them into self-contained operating systems. The big advantage is the possibility to use Windows beside Linux alongside Solaris and any other operating systems without them affecting each other. This leads to a higher flexibility because applications can be installed on the most suitable system. These virtual machines can be taken down for maintenance independently without affecting other systems or moved from one physical hardware to another without adaptations.

In the course of time different vendors produced virtualization products and implemented more features to suit the customer's needs. But all of these products were focused on the need of data centers and none of them addressed the issues on the client computers where desktop applications could not be installed along each other. Certainly, the idea behind solving the conflicts between applications is very similar to the one that forms the basis of virtualization in data centers: isolation and portability. Indeed, the virtualization happens on a higher level because it is not necessary to virtualize hardware components or a complete operating system for applications to be separated.

However, it took until the end of the 1990's for vendors like VMware or Softricity to begin the development of an application virtualization product.

1.2. Types of virtualization

To better understand what application virtualization is and how it differs from the other types, this section will outline the main forms of virtualization.

1.2.1. Hardware virtualization

The concept of this virtualization type is to have a virtual machine monitor (VMM, also hypervisor) that provides a virtual operating platform which enables the execution of several guest operating systems on one host computer at the same time. Thereto, all available resources have to be shared between the host and guest operating systems in a way that guaranties a good hardware utilization. As every guest is isolated from the physical hardware as well as from the other guests, the hypervisor needs to provide a special interface which routes the guest's system calls to the physical hardware. This is also necessary because the virtualized machines are not aware of running on the real physical hardware and are not having direct access to it. This implies a loss in performance but allows the guest operating system to run on many different host computers without any adaptations, as long as the hypervisor is installed.

The hypervisor can be classified into two types: [4]

1. the native (or bare metal) hypervisor that runs directly on the host's hardware, like shown in figure 1.1.

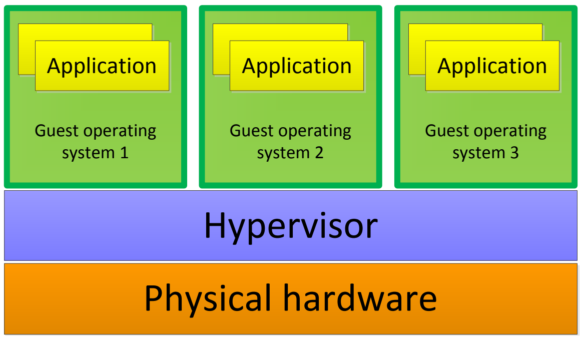

Figure 1.1.: Native hypervisor

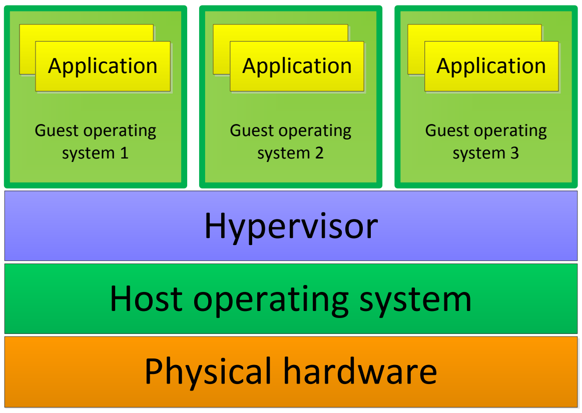

Figure 1.2.: Hosted hypervisor

2. the hosted hypervisor that depends on an installed operating system as depicted in figure 1.2.

In both figures the green border around the guest operating systems symbolizes the isolation that separates all virtual machines from each other as well as from the host operating system (in the case of figure 1.2).

1.2.2. Operating system-level virtualization

This type is one level of abstraction higher, as now the kernel of an operating host system is shared between several isolated guest systems. The overhead is very little compared to hardware virtualization because it is not necessary to provide a virtual set of hardware to every guest. One disadvantage is the fact that every kernel inside a container must be based on the host kernel. Since the latter takes over the hypervisor's job it has to regulate the guest's access to the resources. This implies that it is impossible to virtualize a Windows operating system on a Linux kernel and vice versa because of the kernel incompatibility. Another drawback is that some guest operating systems need a few modifications to become virtualizable. Furthermore, it also requires the host kernel to contain every kernel component (like drivers) the guests will need because there is no possibility to install and load these in the virtual environment [65].

Figure 1.3 shows the virtualization layer that is part of the operating system and isolates the user-space instances, which is implied by the green border.

1.2.3. Application virtualization

Virtualizing applications means to isolate the next higher layer following the operating system-level. So it decouples the application from the operating system and makes it possible to offer programs in a network as Click-To-Run applications [85]. This means that no installation is required to run a program which abolishes conflicts in the system and

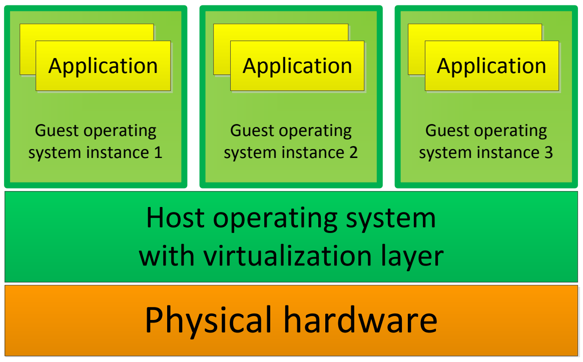
Application
Guest operating system instance 1
Application
Guest operating system instance 2
Application
Guest operating system instance 3
Host operating system with virtualization layer
Physical hardware

Figure 1.3.: Operating system virtualization

makes software deployment much simpler. In addition, the deployment process requires less time because the applications do not need to be configured for different types of hardware. Even a mandatory reboot that a software installation may prescribe is no longer needed and thus saves time for administrators and users.

Like the easy-to-move operating system in the sections above, it is possible to copy an application from one computer to another without causing impacts on the package integrity. This is because the virtual applications contain their own environment with all resources (like files, registry keys, services and settings) they need. As all changes are captured inside the virtual environment, a virtualized application cannot harm other virtualized or natively installed software. Even the operating system with all its core libraries stays untouched.

The concept is depicted in figure 1.4 and shows the integration of virtualized and natively installed applications. Here, instead of operating systems, the applications are isolated, which is symbolized by the yellow border around applications A to D.

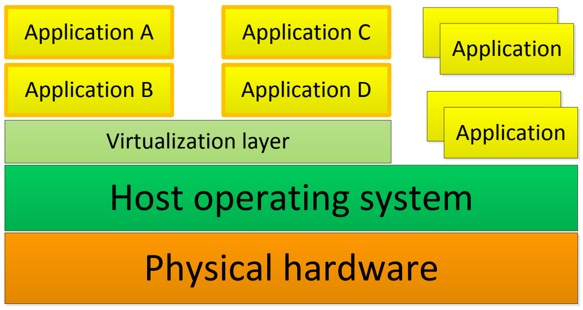

Figure 1.4.: Application virtualization

To achieve and maintain such self-contained environments it needs a virtualization layer on top of the operating system. Its job is to provide a virtual registry and virtual file system which the application assumes to be the real ones. Hence, the application's system calls have to be intercepted and modified, which will be explained in more detail in chapter 2.

As the virtualization of applications does not require to create an abstraction for the hardware components, it is very resource-conserving. Instead, the virtualization layer has to provide a mechanism that is capable of several things [54]:

- An application reads or modifies files that were placed into the file system at installation time. Because of the isolation, these files are not located in the same place on the client. Thus, the layer has to make the application believe that all files reside where it expects them. Thereto, it has to merge the files with the local file system or it has to override existing ones that are different to the ones of the virtual application.
- The same applies to the registry which has to be readable and writable by the application like if it was installed locally. But instead, the changes have to be saved in a separate place to keep the real one clean.
- If an application creates a new file, the layer has to decide if it is a part of the virtual application and thus needs to be captured and isolated as well. On the other side, the file could be a document that needs to be saved to the native file system which means that the layer must not redirect this request to the virtual counterpart.
- Configuration settings that are made by the user have to be user-specific and cannot be placed into the virtual application. Therefor, the layer must create a special data store where all the changes are saved as deltas. Later on, at runtime, the virtual application and the user's adaptations have to be merged again.
- Also changes to environment variables need to be respected and have to be applied only for the virtual application.

Chapter 2 will explain how the virtualization layer works in order to achieve these requirements, while a concrete example is covered on the basis of the Microsoft App-V functionality in chapter 3.

1.3. Benefits of application virtualization

With the virtualization of applications and the concomitant isolation not only conflicts between programs and the operating system are a thing of the past. Also the need for an installation is obsolete and thus the waiting time of an user is greatly reduced because he does not have to wait for program updates and new applications to be finished before he can use the workstation. From a technical point of view this simplifies the deployment of software because a virtual application consists of one single file that includes everything the program requires.

Thus, also the streaming of an application from a server to the client is possible which changes the software deployment from a "push" to a "pull" method where the user only streams the applications he really needs [35].

Furthermore, the isolation also means an abstraction from the operating system which not only enables easier migrations to another version of an operating system [84]. This is because virtualized applications can be recycled by copying them to the new platform without having them repackaged before. Normally, companies have a complex application lifecycle management that involves much time for regression testing. With application virtualization the management is simplified because the potential risk of conflicts is reduced and thus crashes that resulted of such problems are avoided.

Hence, application updates are becoming easier because they do not have to be tested on different machines. With the streaming technology the deployment of a new program version does not require a removal of the older version as with the next launch the program will automatically be streamed in the latest version. Thus, the application only has to be interchanged on the server-side, which means a higher level of security as patches can be applied promptly [83].

Certainly, with the isolation the removal of old version is not necessary anymore. If products have to be available in different versions on the same machine, application virtualization will make this scenario possible because program files, libraries or unequal dependencies cannot conflict with each other.

As all changes to a virtual application are saved in the user's roaming profile, the configuration settings will follow the user from one workstation to another. This is a huge difference to natively installed programs where some of them are not capable of a multi-user environment with network clients and store all settings on the local machine.

Through the separation of the application and the user's adjustments, the support becomes easier for help desks and administrators because the application can be reset to its original state if an user experiences problems with his application [74].

In spite of the virtualization the applications are still able to access devices like USB flash drives, printers, scanners, the local disc and the like. This is an advantage over the other virtualization types because there not all devices can be accessed directly.

1.4. Drawbacks of application virtualization

When a company decides to virtualize at least the programs that conflict with each other, the administrators have to become familiar with a new technology which needs some time and training. They have to learn the differences between their current deployment of software and the way how application virtualization works. Furthermore, the new product must be integrated into the existing network which may require adaptations and extensive tests to servers and clients. In complex networks it might be a good idea to do a pilot experiment with only a few workstations and briefed users to become acquainted with the product.

Moreover, the problems that may arise while creating a virtual application can differ from the standard software packaging process. Some vendors might not support those problems because the application was not designed to run in virtual environments. Thus, in such cases, the problem solving can become more complicated because of the execution in a virtual environment which will cost more time or even result in the conclusion that the application is not virtualizable.

Additionally, another product means to pay more money for licenses, which depends on the amount of clients that will run the virtual applications [75]. So it must be considered whether application virtualization is a benefit for the company or if the costs outweigh the advantage.

If applications are streamed from the server to the client on demand, the server might become a single point of failure when it is not available [48]. Microsoft App-V caches the applications locally on the client which allows the execution of a program even if the server is not on-line, as long as it is still cached. Other vendors do not offer such a fallback solution and require high availability on the server-side. In some smaller networks this might not be the case, so the network administrators have to plan beforehand which virtualization product suites most of their needs and think about products that offer stand-alone modes.

1.5. History

To understand why there is a growing demand on application virtualization it is necessary to clarify the root cause of today's application conflicts.

1.5.1. The time before the Windows Registry

The first versions of Microsoft Windows installed applications into separate directories that contained everything the program needed to run. Hence, all configuration parameters were put into configuration files (recognizable on the filename extension `.ini`) and dependencies (such as libraries) had to reside in the program's directory [88]. This not only enabled administrators to just copy the directory from one machine to another without missing mandatory components. It also made it possible to have various applications using different versions of the same software and use differing settings.

So there was a natural isolation between the programs on the file system level but shared dynamic link libraries (DLLs) could cause incompatibility problems of running programs. This is because there was only one shared address space and, therefore, no chance to use different versions of the same library for differing programs [31].

The configuration files also had the drawback that they were not designed to support user-specific settings which meant for multi-user environments that end-users had full access to the application and were able to override the settings of other users [29].

1.5.2. After the release of the Windows Registry

With the release of Windows 95, Microsoft established the concept of the `Windows Registry` as a central place for saving configuration settings. This allowed programmers to store user-based settings in the user's part of the registry, share settings between applications and make use of system options [49]. The downside of this new design was – and still is – that applications can override each others' settings if they use the same paths in the registry. This can lead to a significant problem when a program consults the registry to determine the location of a component and loads a different version of the required library as it had registered in the installation process. The consequence of this conflict can be one or more damaged application installations or – in the worst case – an instable or even unusable operating system.

Furthermore, since Windows 95 every process is executed in its own address space which allows an isolation of its data in memory as long as they are not shared with other processes [29]. Certainly, the file system is still not able to handle programs that override or require different libraries in the system folder.

This problem is commonly known as "DLL Hell" which is a Windows-specific phrase, but the general term "dependency hell" also applies to other operating systems [88]. The root cause is that Windows uses one single namespace for all files and thus it is impossible to have two different versions of one DLL installed in the system32 folder if they need to have the same filename [81].

Normally, an issue with file conflicts is solved by installing the different versions of a software into separate locations, for example, onto distinct partitions. But this would not solve the whole problem because system components are always installed into the same destination and let the latest installation override the existing libraries. The same applies to the Component Object Model (COM), which allows interprocess communication and the creation of dynamic objects. As all COM classes are saved in the Windows Registry with a specific name and there also is only one namespace there is no possibility to register a COM with a particular name in more than one program.

Not only the installation of a program can lead to a system instability, also the uninstallation is very risky. This is because a software may remove all the files it placed into the system during the installation phase, regardless of other programs meanwhile depending on them.

But there is also software that does not cause problems during the installation and still faces conflicts that prevent their execution. Such incompatibility problems are called "dynamic conflicts" because they only appear at runtime and under certain circumstances. An example for these is the use of the same mutual exclusions or semaphores [5].

Without application virtualization such conflicts can only be solved by installing the programs onto different machines, or at least into different operating system instances. This approach has several downsides for both, the administrator and the end-user. For the administrator the management of additionally machines means a more complex software deployment and additive sources of error. On the other side, the user has to switch between different machines or operating systems in order to use the swapped programs which decreases his efficiency and makes it more complicated.

Using terminal servers instead reduces the number of distributed machines and thus saves some support time. Nonetheless, the problem of conflicting applications stays the same if a remote desktop infrastructure is used. Hence, these applications have to be separated into different remote desktop sessions and users have to choose which session they need to open in order to use a specific application [37].

With Windows Vista, Microsoft introduced a file and registry virtualization concept that is part of the new User Account Control (UAC) technology. It ensures that only administrators have the right to write into the application folder in `C:\Program Files`. Thus, if an application is executed in the context of a standard user it cannot write to the application folder because of missing permissions. Instead, the UAC redirects the file or registry operations to a specific location within the user's profile which is called the `Virtual Store` [22]. This technique must not be mistaken for application virtualization as it only protects the application from being changed by normal users. Certainly, a virtual application is protected from user-specific modifications, too, which is based on the same idea.

With the Virtual Store, Microsoft reduced the likelihood that users override the settings of other users inside the file system or registry but the issue of conflicts between applications is still there. Also, it does not protect the program directory at the installation time of another application as the setup runs in a privileged mode and thus the installer can override files and registry keys of other programs.

1.6. Structure of this thesis

The structure is as follows: the next chapter explains the technical background behind application virtualization and the functionality of a virtualization layer. Thereto, a short introduction is given into the Windows architecture and the technique of intercepting function calls in order to show how the virtualization layer organizes the redirects.

Chapter 3 will introduce the virtualization product of Microsoft in more detail. Thus, the components of Microsoft App-V and its functionality as well as its advantages and drawbacks are outlined here. The addressed features will be picked in the following chapter to demonstrate if and how they can be used in networks without Microsoft's server infrastructure.

The practical part of this thesis is represented in chapter 4 which contains the integration of App-V into Munich's school network. There, the concept of application virtualization is intended to solve conflicts between complex software products that need to be installed on the workstations without problems. Thus, App-V is part of the upcoming Windows 7 pilot experiment which will prove whether it can fulfill the needs or not.

As there are several other application virtualization products beside App-V that partly take a different approach, chapter 5 compares three other products to App-V. Some of them have interesting attributes that can be of interest for companies where Microsoft's proposition is not compatible with the network infrastructure.

The last chapter sums up the previous sections and estimates whether application virtualization is appropriate for everyone or if there are scenarios where the commitment is in dispute.

2. Background

As mentioned earlier, in versions starting from Windows 95 every program has its own address space and thus cannot read or write the data of other processes. Given that it is impossible to implement the virtualization layer as a piece of code that manipulates the virtualized program's library at runtime by just changing a few pointers in memory or by replacing some functions with adjusted system calls.

The simplest way would be to modify the source code and change the original function and point it to a modified version inside the application programming interface (API) of the virtualization layer. Obviously, this approach is very time-consuming, requires human interaction and cannot be done at runtime because the program has to be recompiled. As most of the applications are licensed as proprietary software, there is no possibility to adjust the source code in this way [50].

Hence, the virtualization layer has to use another technique of making an application independent from the operating system at runtime. The functionality of intercepting the application's function calls is described in this chapter.

2.1. Architecture of Windows

To understand where the program's function calls occur and at which level the virtualization layer should be implemented to intercept those, it is best to have a look at a simplified architecture of the Windows operating system. The basic items are shown in figure 2.1 and span user as well as kernel mode.

Generally, applications run in user mode and perform operations on system services that reside in kernel mode. This, for example, can be a system call to open a file for writing. Thereto, the application normally uses a function inside the kernel32.dll library that is part of the `Windows API`. This subsystem also provides libraries like user32.dll (for the user interface) and gdi32.dll (for the usage of graphics devices) which are commonly used by almost every program. Then, if the function is completely implemented inside the Win32 API and does not require any additional services outside the Win32 subsystem, it will be executed. Otherwise, the system call has to be handed over into kernel mode. This transition is done by the `Native API` which is a largely undocumented library that was compiled into Ntdll.dll inside the system32 directory [78]. It offers many functions that are used as a gateway to access the Windows Executive services which are contained in the `Kernel` (ntoskrnl.exe). With it, the application can make use of the I/O Manager, Cache Manager, Process Manager, and some other subsystems [77].

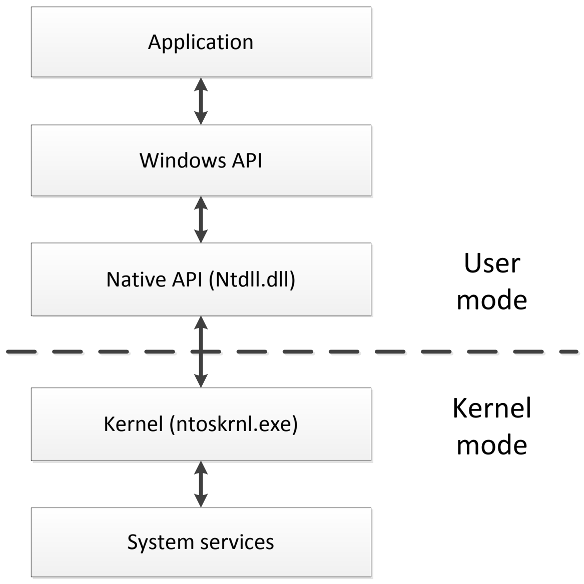

Figure 2.1.: Simplified architecture of Windows

2.2. Placing the virtualization engine

To intercept a system call it requires a virtualization engine that provides functions for capturing an application's system call and manipulating it where needed. As there are several libraries involved which are passed by the system call on its way to the Windows Executive services, it has to be considered where the interception may happen.

First, it has to be decided if the engine will run and intercept in user or kernel mode. The choice is important as it will determine how the layer has to be implemented and with which permissions it will do the filtering.

2.2.1. Interception in user mode

When the interception is done in user mode the virtualization engine has to be placed over the Native API layer like shown in figure 2.2. This is because there are some applications that are not using the Windows API and instead call the functions of the Native API directly. There are a few reasons why the layer in between may be omitted:

- The Native API exports functions that are not offered by the Win32 subsystem and thus need to be accessed directly [78].
- Some programs are started before the Win32 subsystem is available and hence have to hark back to the Native API [79].
- Using native APIs is sometimes a little less overhead than calling another abstraction layer prior to that.

Thus, placing the engine above the Windows API would imply that some function calls could not be intercepted and the virtualization of these programs would fail.

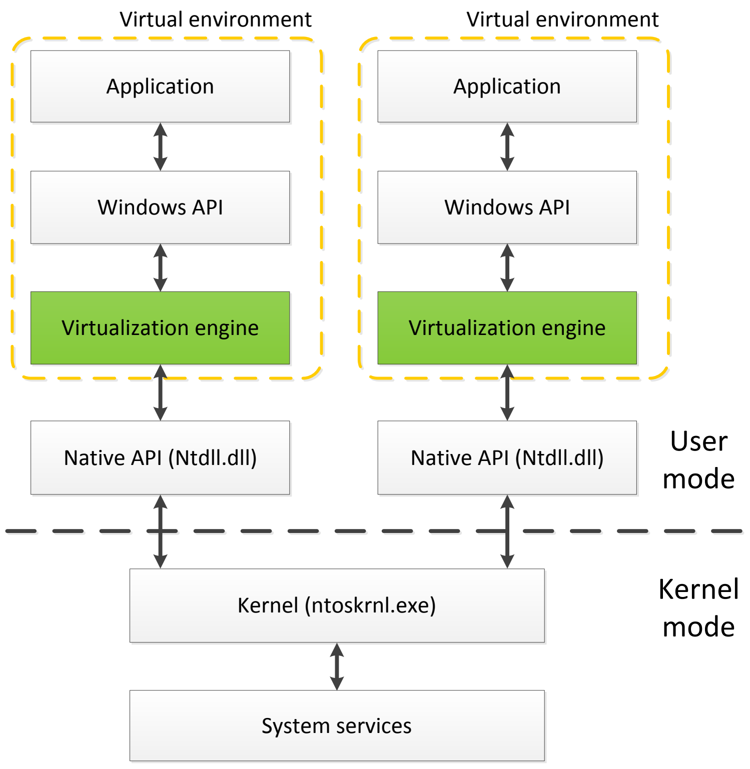

Figure 2.2.: Interception on the Native API level

One advantage of the user mode strategy is that the virtualization engine does not have to filter between system calls of different applications as it only receives the ones of the application that it built the virtual environment for. This enables virtualization products to include the agent that is responsible for the virtualization inside the virtual package. Thus, it is not necessary to install a client on the end-user's machine and it is even possible to generate one portable executable that contains everything to run the virtual application [6]. Another advantage is that the virtual application is running with user privileges and thus does not require administrative rights nor can it break the system when it crashes [83].

2.2.2. Interception in kernel mode

In contrast, if the engine operates in kernel mode it can intercept all system calls before they reach the Windows Executive services. This scenario is shown in figure 2.3.

The advantage is that it is easy to loosen the isolation of applications and let them share the same virtual environment. This enables to create dependencies between packages, for example, when working with middleware or plug-ins.

A drawback is that the agent that builds the virtual environment has to be installed natively on the machine to gain the required privileges for the interception in kernel mode. This implicates that an error of an application that runs in kernel mode can lead to a crash of the agent which would have an impact to all running virtual applications. It even could harm the whole operating system.

Furthermore, the virtualization layer will capture system calls from all applications and has to filter the processes in order to treat every call correctly [6].

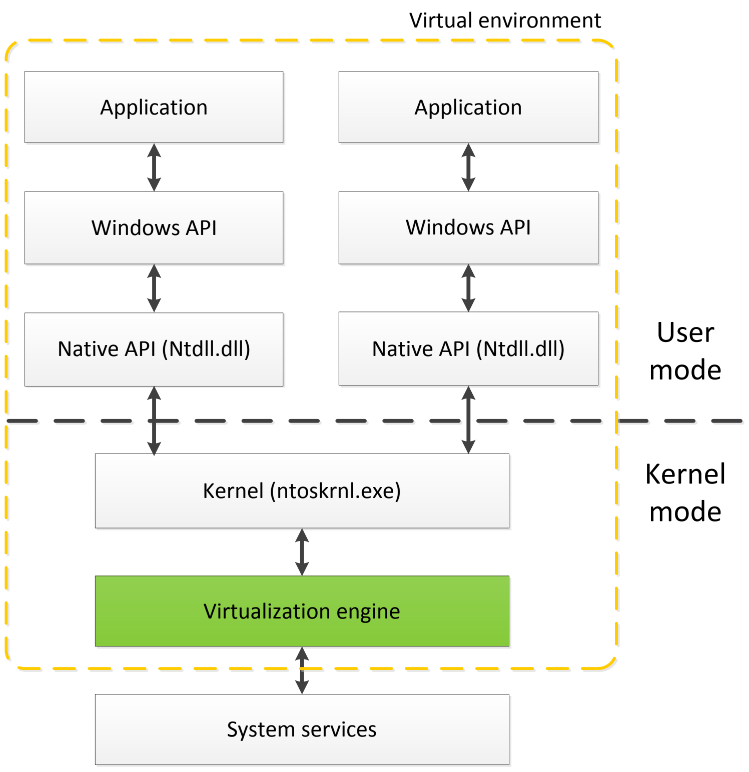

Figure 2.3.: Interception in kernel mode

Depending on the problems that a virtualization product should solve, one of these two modes will be chosen. But it is also possible to combine both ideas and make the location of interception dependent on where the application needs to be executed. This hybrid approach was implemented by Microsoft App-V while VMware ThinApp decided to intercept in user mode only [28, 57].

2.3. Functionality of a virtualization engine

In order to decide which function calls to intercept and which have to be handled as usual, the virtualization layer needs some kind of rules. These are generated at the time when the application's installation is monitored by a special software that is shipped with the virtualization product. This program will collect all the files and registry keys that the installer creates or modifies. Then, at the application's runtime the virtualization engine uses the information to ensure that function calls are only modified if they request a path inside the virtual file system or a key inside the virtual registry.

When the engine runs in kernel mode, it must additionally check which package needs to be consulted to find the rules, as it will receive function calls from several running applications that belong to different packages. This can be achieved by maintaining a list that assigns a running process to a package and then looking up to which package a process belongs to.

So, the virtualization layer has to intercept and redirect requests to the file system and registry to virtual counterparts that contain the files and keys that belong to a certain application. Like shown in figure 2.4 the engine also must ensure that requests to the native components are still possible in order to use files in "My Documents" or locally installed middleware or drivers. Certainly, the isolated application must not have write access to the native components to ensure that it cannot change or break the system. But saving a document is one exception where the application must be able to write to the file system.

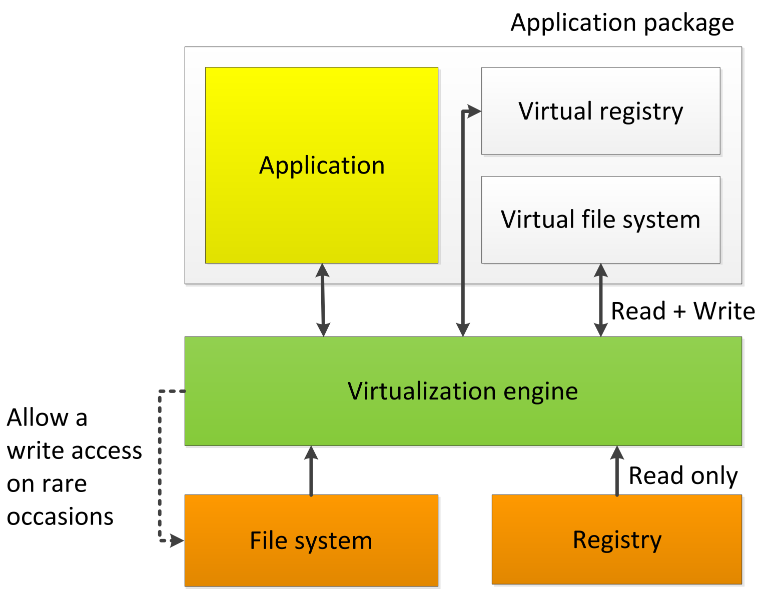

Figure 2.4.: Redirection of requests by the virtualization layer

Furthermore, application-specific environment variables must be provided inside the virtual container. Thus, the software which creates the virtual application must collect the variables that have been set by the installer. One way to do this is to export the list of environment variables before the installation and after the monitoring phase. Then, the changes between both lists can be taken as the virtual application's environment variables [51]. At runtime, the virtualization layer creates an environment block for the new process in memory and passes the pointer as an argument to the `CreateProcess` function [12].

2.4. Windows API hooking

This section will explain how function calls are intercepted in general and it shows the functionality of an API hooking library that is sold by Microsoft. Furthermore it will describe a way for file system and registry filtering that is easier then the API hooking approach.

2.4.1. Basic mechanism for intercepting function calls

Instead of modifying a program's source code or manipulating its code segment in memory, the layer has to intercept specific system messages and handle these on his own where necessary. This is done by a technique called `hooking` which allows the injection of special DLLs into the context of a running program [70]. Typically, this method is used to extend the functionality of a function or to insert code for debugging purpose. But it is also possible to inject a hook that modifies the return value of a function, which is the basic idea behind the virtualization layer.

The user-defined library that will be injected into the target process must contain a replacement for the target function which has to be hooked. Therefor, the DLL can be placed into a special registry key (`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs`) that will be attached to every process that uses user32.dll [33]. The downside of this approach is that even not virtualized applications will get the library attached. This is because user32.dll includes standard elements of the Windows user interface which is imported by most of the programs that implement a graphical user interface.

Instead, the virtualization layer must provide a functionality that creates the target process or opens it to get a handle. After it has an entry point to the process, some memory is allocated in the target process and the name of the replacement DLL is written to it. Then the launching function has to create a remote thread and pass the DLL's name as an argument to get it loaded by the target process [63]. After that, every call to the target function will execute the replacement function that has been injected with the DLL.

In order to hook the Windows registry, the virtualization layer has to implement functions like `RegOpenKey`, `RegCreateKey`, `RegSetValue`, and `RegDeleteKey`, but there are many more functions that need to be observed to find out if all of them have to be implemented [26]. On the other hand, to intercept function calls in the I/O stack, the layer has to hook methods like `CreateFile`, `ReadFile`, and `DeleteFile` [14]. These will override the existing functions and replace a path with the new location of a file in the isolated area where the virtual application's files are stored. Thereto, the before mentioned list of collected rules is used to find the storage location of a requested file. If the file is not part of the virtual application, the layer can pass the request on to the operating system which will return the file from the native file system.

2.4.2. Microsoft Detours

As an example for a library that makes API hooking easier, Microsoft Detours will be introduced in order to explain how the injection of a replacement function works.

Detours is a product by Microsoft Research that can be used to intercept Win32 functions on 32-bit and 64-bit machines dynamically at runtime [76]. When a function is hooked, Detours modifies the target function in memory without having to touch the binary on disk. Therefor, it has to locate the original function (also `target function`) in memory and replace the first 5 bytes with an unconditional jump instruction to the user-defined `detour function` that implements the new functionality. Prior to that it copies the target function's instructions to a newly allocated space (so called `trampoline function`) in memory to preserve them. This way it is either possible to replace the target function completely by the detour function without calling the original code or to jump into the trampoline function afterwards, which will additionally run the original function [50].

The insertion of a detour is shown in figure 2.5. When a hooked target function is called by a source function (step 1) the detour function will be executed and behave like described above. If it replaces the target function, the detour function will return to the source function at the end (blue path in the figure). Otherwise, it calls the trampoline function (step 3b) and executes the original code (equates to the orange path in the figure).

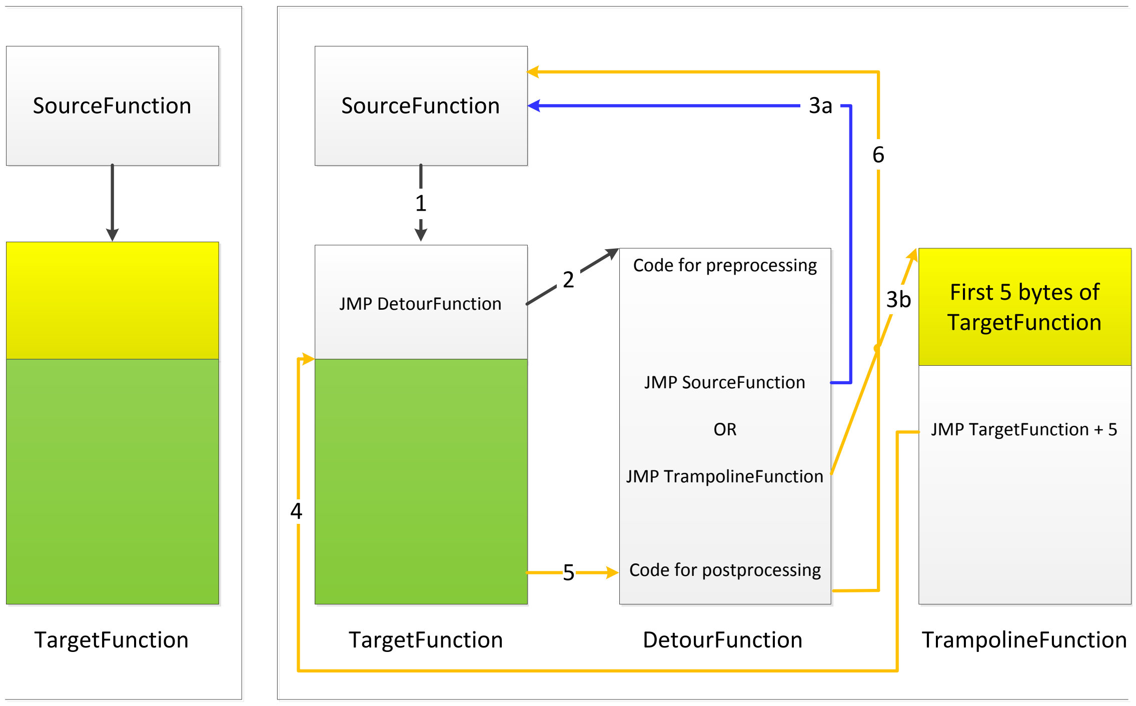

Figure 2.5.: Situation before and after the insertion of the detour

2.5. Intercepting requests to the file system and registry

In order to filter or intercept requests to the file system, Windows offers an easier way as hooking the Windows API, which is also documented. Thereto, one can write a `File System Filter Driver` that is registered in the operating system and will be executed in kernel mode. The so called `minifilter` is linked to the `Filter Manager` which is responsible for intercepting the system calls to the file system and calling the registered minifilter. Thus, the filter driver is able to monitor all requests between the I/O manager and the file system like shown in figure 2.6 [16, 17].

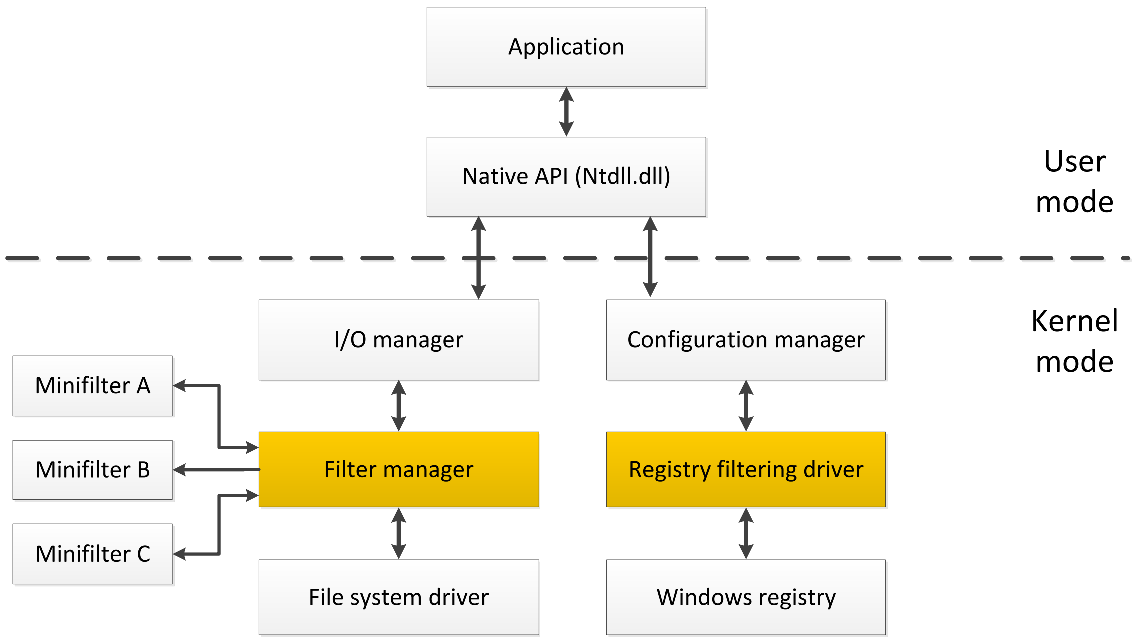

Figure 2.6.: Positioning of the file system and registry filter

Since Windows XP the Filter Manager is part of the operating system and minifilters can be written by using a special driver kit which is provided by Microsoft [15].

For the interception of the registry, Windows provides a `Registry Filtering Driver` which can be used to register a callback routine that will be executed at every registry operation before the request is handled by the `Configuration Manager` [18].

3. Microsoft Application Virtualization (App-V)

Microsoft's product for application virtualization aims to remove conflicts between programs and thus to reduce the costs for compatibility tests. Therefor, they provide a set of different software components that allow network administrators to generate virtual applications and publish them dynamically to the users through a built-in deployment infrastructure. As the program settings are saved separately, the user's environment becomes independent from the machine and thus the IT department can react to changes in the business process very fast. Hence, also the migration from one operating system to another becomes an easy task because the virtual applications do not have to be installed or adapted to the new system.

This chapter will explain Microsoft's product in more detail and outline its advantages and drawbacks.

3.1. History

The development of App-V was started by a company called SoftwareWow in the late 1990's and early 2000. At first, they sold application streaming software to their customers by hosting all applications in their own data centers. These times, the applications were not virtualized but have already been streamed to the customer's computer over the Internet where the program was executed without an installation. Thereto, they provided a client that had to be installed on the end-user's machine once. Then, to start applications, the customers had to grab them from special websites that had a partnership with SoftwareWow. The streaming was initiated by clicking on an application and executing a JavaScript routine that handed the control over to the locally installed client. Later on, they realized that more and more customers experienced both conflicts with the applications and problems with the streaming over the Internet. Thus, they began to implement a new product called SoftGrid, which included application virtualization and the caching functionality on the end-user's client. Furthermore, the streaming feature was improved by downloading the necessary parts first, in order to start the program as quickly as possible [69].

This happened around the end of 2000 and early 2001, and came along with the company's rename to Softricity [68]. In the year 2006 Microsoft acquired the company to extended their virtualization product family with SoftGrid [32]. With the release of version 4.5 Microsoft changed the product name to Microsoft Application Virtualization (App-V), but there are still remains of the old name, like in the registry, to ensure backwards compatibility.

Today, App-V is part of the Microsoft Desktop Optimization Pack (MDOP) and hence only available to subscribed Software Assurance customers.

3.2. Functionality of App-V

As already mentioned in the introduction chapter, Windows uses one single namespace for all files. To circumvent the conflicts that arise by this scenario, App-V provides a separate namespace for every single package. At runtime App-V layers it on top of the local namespace to build the virtual environment. However, this is not enough as no program is capable of using more than one namespace, so the virtualization layer has to combine the local and virtual namespace to one single environment. The obvious problem that needs to be solved is to present the correct file and registry key to the application when it sends a request. This includes the necessity to hide a local file inside the virtual environment when it is deleted by the virtual application.

The virtualization layer is integrated into the App-V Client and spans user and kernel mode as illustrated in figure 3.1.

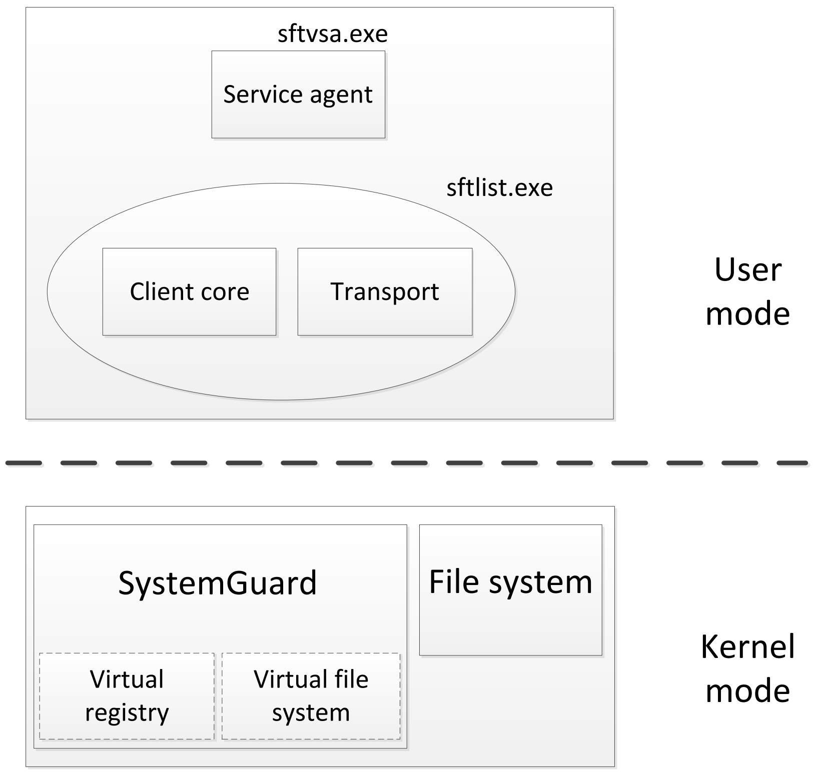

Figure 3.1.: Virtualization layer of App-V

A Windows service called the "Listener" is running in user mode and contains an orchestration engine besides a transport component. As App-V consists of loosely coupled components, the orchestration engine (which is also the `client core`) has to manage the various subsystems. The `transport` component handles all network traffic, both the virtual application's one and the traffic that is part of the streaming process. At installation time of App-V the Listener is registered as a background service named "sftlist.exe" and is started by the operating system at boot time.

Also, the `Virtual Service Agent` (VSA) is installed as a process called sftvsa.exe that starts at boot time. This component handles all services that are part of the virtual application and thus running inside the virtual environment. As the services cannot be registered in the native operating system, the service agent is responsible for controlling the communication between processes and services inside the virtual environment [80].

The virtualization engine called `SystemGuard` and a special file system named `App-V volume file-system` [38] are executed in kernel mode. The latter is handling paging I/O operations for streaming and thus sends requests over the network to fulfill the paging. Furthermore, it offers a copy-on-write feature that comes into play when a user modifies a file inside the virtual environment. Such a change triggers the file's duplication and subsequent storage in the user's delta file. Afterwards, the modification is written to the copied file and henceforward the user's delta is used to fulfill the request for the file handle. Although it is a custom-made file system, it is registered as usual in the operating system and stores all the files that are part of the virtual application.

On the workstation the drive is created at the installation time of App-V and from then on standardly associated with the drive letter Q, but it is left free to be chosen. The drive is also present on the machine that creates the virtual applications. There it plays a role in the installation phase of a program when it comes to choose an installation directory for the application. Here, the drive Q is preferred over the system drive because file requests to the Q drive do not have to be redirected by the virtualization layer at runtime. This makes the program startup a little faster and requires less computing time. The direct access of an application to files inside the Q drive is shown in figure 3.2.

Certainly, the Q drive is more like a virtual drive that contains all the files and registry keys an application needs. Thus, in order to protect the so called "golden image" of an application, the drive can only be accessed by the App-V Client. So it is not possible to browse the drive within the Windows Explorer although the App-V Client is allowed to write to the drive in order to place new applications or remove the ones that a user is not allowed to use. A virtual application itself has only read permissions to its own package folder or to root folders of other applications that have been marked to run in the same virtual environment. The delta file that contains all the changes that a user makes to the application, is stored in the user's profile directory. Thus, the golden image stays untouched and all configuration settings can stick to a user if he switches the workstation by using roaming profiles that are saved on a server.

The SystemGuard contains a `virtual registry`, `virtual file system` (VFS) and a `virtual environment manager`. Contrary to the App-V volume file-system, the

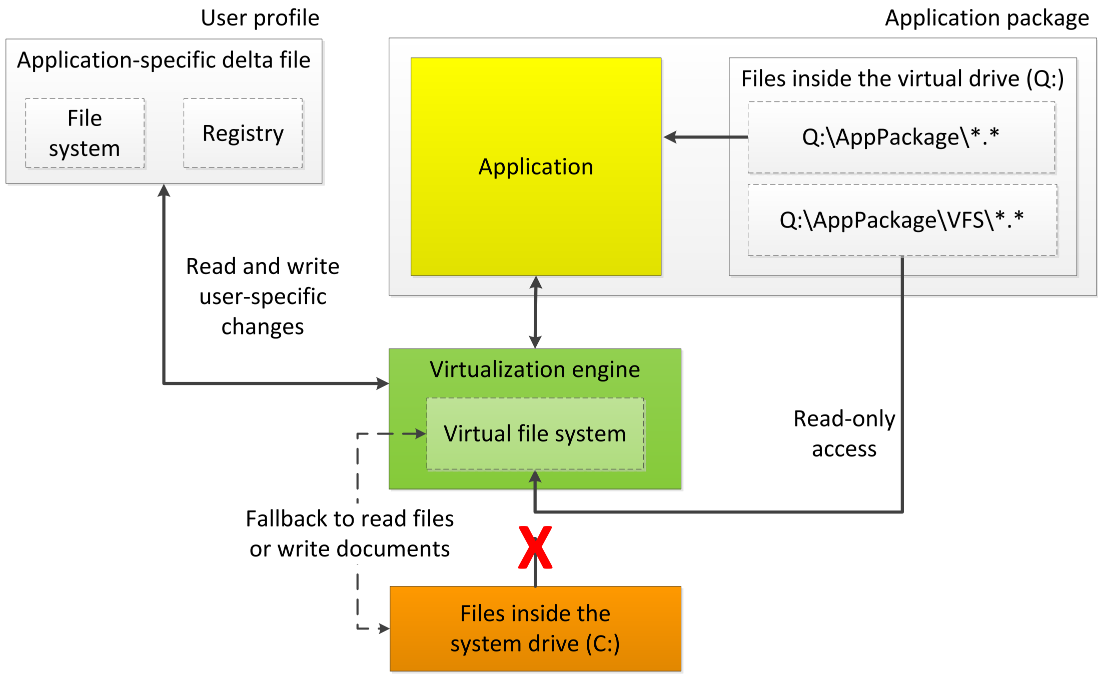
User profile
Application-specific delta file
File system
Registry
Application package
Application
Files inside the virtual drive (Q:)
Q:\AppPackage\*.*
Q:\AppPackage\VFS\*.*
Read and write user-specific changes
Virtualization engine
Virtual file system
Read-only access
Fallback to read files or write documents
Files inside the system drive (C:)

Figure 3.2.: Direct file requests to the virtual drive and redirected ones to the system drive

virtual file system does not store any files but is responsible for the composition and redirection of files, directories and registry keys. This feature is necessary because many applications install files to the system drive, although the installation folder was chosen to be the virtual drive of App-V. At runtime those files have to appear where the application expects them. Thus, the virtual file system filters the file system traffic and redirects requests to such files to the VFS folder in the virtual drive Q like shown in figure 3.2. Only if the request does not find an entry inside the virtual file system catalog, it is passed to the operating system and processed there. This way the application can use a natively installed middleware software or drivers besides opening documents from the real file system.

```
Q:\
  AppPackage\
    VFS\
      CSIDL_WINDOWS\
          foo.dll
    application.exe
    osguard.cp
```

Listing 3.1: Directory structure of the virtual drive

Listing 3.1 shows the directory structure of the virtual drive. For every application package exists one directory in the root of the drive. The package folder contains a file called `osguard.cp` which includes the virtual file system and virtual registry mappings that define which paths and keys need to be redirected to the VFS folder. Furthermore, all files that have been installed to the Q drive are located there, for example, the executable file of the application. Inside the VFS folder all files are stored that could not be installed to the virtual drive and are the target files for the redirects.

If, for example, the application sends an open request to the file `C:\Windows\foo.dll` the virtualization layer captures the call and determines whether it has to redirect it. Given that the file belongs to the application it will be part of the VFS mappings inside osguard.cp and thus will be redirected to `Q:\AppPackage\VFS\CSIDL_WINDOWS\foo.dll`, at which the directory name CSIDL_WINDOWS depicts a system-independent way to identify special folders [13].

The redirection is done by a minifilter called `sftredir` which is registered in the Filter Manager like described in section 2.5. So, the whole application resides in the virtual drive Q and all files and registry keys that belong to the program never leave that drive, which ensures its isolation.

Running a virtualized application is very lightweight, because the redirection is not done at every read and write access but only once for the first access of a file. Thereto, it intercepts the function call and replaces the parameter that contains the file path and replaces it with a new one that points to the VFS folder or to the virtual registry. This replacement is done by using the metadata of the virtual file system inside the osguard.cp file [67]. Afterwards, when the file handle is established, the I/O works without additional redirects. This way an application can access the hardware like if it was installed locally, which is a huge difference compared to hardware virtualization. Thus, it is no problem to virtualize programs that utilize the CPU or graphics card a lot, which is a plus for application virtualization.

The virtual environment manager maintains the virtual environment and instructs SystemGuard to put the application's process inside this environment. It additionally controls the lifetime of the virtual environment and ensures that all changes are saved and the environment is shut down when there is no process left inside it [81].

3.3. Constraints of App-V

However, there are a few things that cannot be achieved with App-V: applications and components that are part of the operating system and thus are integrated into the kernel are not technically supported to be virtualized. This means that driver, COM+ and COM DLL surrogate virtualization is also not possible with App-V. This further includes Microsoft Internet Explorer which is highly integrated into Windows and thus is also no candidate for virtualization [11].

The virtualization of device drivers is not supported by App-V [41] because an isolation would make them unusable and thus they have to be integrated very tightly into the operating system by an virtualization layer. This means, that a driver cannot be that isolated as normal virtual applications and thus it makes no big difference to a native installation process. Additionally, it is even more problematic to virtualize device drivers because if the interception and redirection is not done right, the driver could crash and cause a system failure because of its execution in kernel mode.

Even though Windows services are supported, services that require to be started at boot time are not. This is because of the isolation as it is no longer possible to register the service in the usual way. Also a virtualized service cannot exist without the corresponding virtual environment. Thus, it is required to launch the service like normal virtualized applications and have it started after the virtual environment has been created [41]. A fairly simple workaround to this problem is to put a virtual application's shortcut item into the startup folder of Windows' Start Menu. This will launch the application, build the virtual environment and automatically start the service if it was configured to behave this way.

Another thing to mention is that App-V is not a security product like, for example, a sandbox software. As the virtual application is still able to make changes to the local file system, a computer virus would cause the same harm like in unvirtualized circumstances. This behavior of local interaction is necessary in case a user saves a document or browses the file system to open a file, the virtualization layer has to bypass these operations. It would not make sense to decouple the application completely from the native file system because it is very important that the user experience is the same as if the application would be installed natively. Furthermore, virtualizing not only the necessary parts of the application would result in a huge overhead that could be compared with an operating system virtualization. Thus, it only redirects file accesses that have been monitored at installation time and are part of the virtual application.

Although App-V generates portable applications that are abstracted from the operating system, it does not offer a compatibility layer. This means that if a program is not designed to run under a specific version of Microsoft Windows, it is very likely to have the same incompatibility issues when it runs virtualized. But if a program is supported for Windows XP and upwards, it is possible to build the virtual application on the lowest version that is used in the network and run it on all other versions above. This is because the App-V Client transforms the virtual application's components accordingly to the Windows version at runtime. Thus, it does not matter if the application package was built on Windows XP and is afterwards executed on Windows 7 as the App-V Client will translate paths like `C:\Documents and Settings\username` to the appropriate counterpart (`C:\Users\username`) [61].

Hence, App-V cannot be used to run 16-bit applications on a 64-bit version of Windows because on these platforms Windows does not support such applications [7]. Certainly, 32-bit applications that have been virtualized can be executed with the App-V Client on a 64-bit machines because of the WOW64 (Windows-32-on-Windows-64) subsystem which offers the compatibility layer.

3.4. Components

3.4.1. Sequencer

The App-V Sequencer is a graphical component of the App-V software package that monitors the installation of an application and captures the changes that are made to the system. This includes the creation, modification and deletion of files and directories in the file system, the edits that happen to keys in the registry and the registration of file type associations. Furthermore, it captures installed services and file permissions besides changes to environment variables. The term “sequencing” means the process of creating the virtual environment.

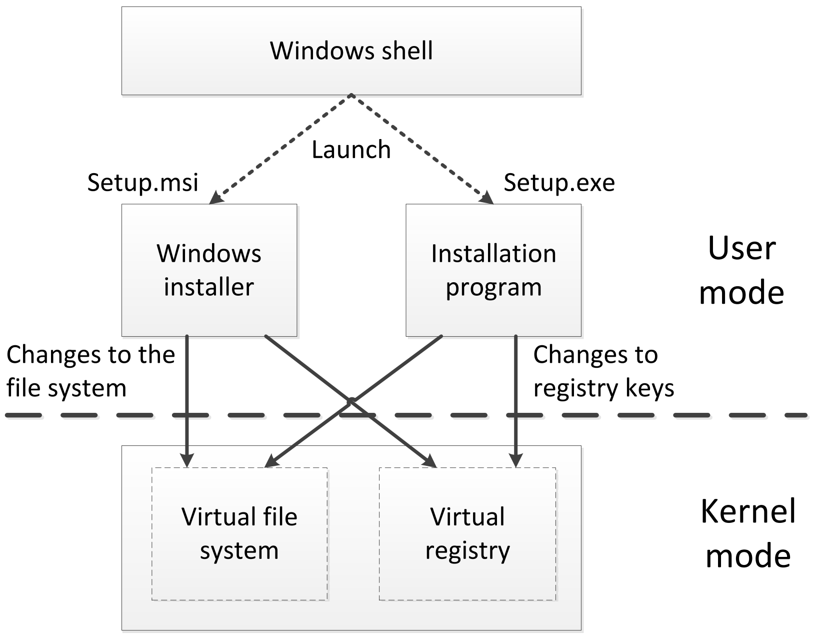

Figure 3.3.: Monitoring an installation

Figure 3.3 illustrates the installation of an application and what the App-V Sequencer must be aware of. Instead of doing a snapshot before and after the installation and taking all the changes that were made in between, the Sequencer monitors every single process that is part of the setup. This ensures that changes not relevant for the application’s virtualization are not captured. Thus, the Sequencer also relies on the sftredir minifilter that will filter all file system traffic and redirect calls for special processes. In order to monitor only the relevant processes the process manager component of the App-V Sequencer keeps a list of process identifiers, including the process ID of the Windows Explorer (explorer.exe), the command prompt (cmd.exe) and the Service Control Manager (services.exe). All of these have to be monitored because they may be a starting point of an application’s installation. The process manager also ensures that child processes are taken into account in order to be able to capture the setup process that was launched by the Windows Explorer or the Windows Installer service, too [80].

At the end of the sequencing process the software builds the before mentioned namespace. Thereto, it generates the `virtual application file` which holds all the metadata that are required to construct the namespace. This file has a .SFT (Softricity) file extension and includes files, registry keys, fonts, environment variables and the information about which files and registry keys form special components like services [55]. Optionally, it creates an installation package (also known as “MSI file”) for networks that do not have Microsoft’s infrastructure or servers to offer another solution for alternative deployment systems. This file registers the virtual application on the App-V Client and contains amongst others the instructions on where to create shortcuts and which file type associations to register.

Arrangements to the sequencing system

It is best to use the Sequencer on a clean machine which means that no disturbing programs, like antivirus software or indexing services, should run in the background while the application’s installation is monitored. This is because such programs are changing the system while they are executed and these changes would unnecessarily be captured into the virtual application. For this reason it is recommended to do all sequencing and testing in separate virtual machines that have a minimal setup. Therefor, products like Microsoft VirtualPC or VMware Workstation are a good choice because they also allow to reset the virtualized guest operating system to a specific snapshot and thus save an amount of time when it comes to restore the machine to a clean state. As an alternative it is possible to work with disk images that are used to reset the system.

Moreover, the machine’s setup should not differ from the productive environment where the application is going to be used. This does not mean that the hardware has to be identical but the initial software installed on the clients should be equal as it simplifies testing a lot when the conflicts arise before the application is rolled out.

As mentioned earlier, App-V uses a virtual drive labeled as Q on the client from which the application is executed. On the Sequencer there must also be a Q drive in order to install applications into this location. But here it is a real NTFS formatted file system that does not differ from other drives. The only specialty about it is the fact, that App-V uses the Q drive as a temporary folder for all files, including the virtual file system and the files of the application if the installation path was changed to the Q drive. So in the monitoring phase it redirects everything to this location which is also known as the App-V mount point drive [11].

Preliminary considerations before virtualizing a program

The whole process of building a virtual application is guided by a graphical user interface which makes it quite simple for beginners to start with. However, before a program is virtualized it is a good idea to become familiar with its installer, the possible configuration settings and the application’s documentation. Also searching the Internet for tutorials on virtualizing the software is recommendable as some applications need very specific modifications to get them running inside the virtual environment. This will give a first hint on how complex the sequencing will be and if the program is a candidate for a successful virtualization at all.

Thus, the program should be installed at least once without running the Sequencer and examined on where it stores user settings and how it integrates itself into the system. This information is important for defining the exclusion list which needs to be adjusted for every program. Some installed files need to be excluded if they have to reside outside the virtual environment. Such is the case, for example, with user profiles that have to be saved in the native file system to allow the usage even with natively installed applications which would be impossible otherwise.

As the Sequencer allows the modification of captured files and directories after the package was built, it is easy to remove included and unneeded data items afterwards. It is even possible to add excluded files manually afterwards if one realizes that the virtual application file is missing important components. But normally, when files are missing, the program does not work properly and the complete sequencing process has to be redone with an adapted exclusion list.

Sequencing process

Once the application has been tested enough, the sequencing process begins with ensuring that all unnecessary background services are stopped. Therefor, the software checks if Windows Update and Windows Defender are disabled. It also informs the user of a previously made installation to indicate that there might have been unwanted changes to the system which could cause conflicts afterwards. This normally happens very rarely, for example, if the virtual machine has not been reset to a clean state after the last sequencing.

If all requirements are fulfilled, one can specify if the setup file should be launched by the Sequencer or executed manually. The latter can be useful if the setup needs to be started with special parameters (like a silent installation). After defining the package name, the monitoring phase begins and the Sequencer captures all changes that are made to the system. It even takes excluded directories into account but only logs such changes to a protocol file. This can be quite helpful if the virtual application does not work as expected, because it provides the information about whether the exclusion list contains paths where the installer saved files to and which have thus not been captured into the virtual package.

When the installation is over, the monitoring phase has to be stopped and then all changes that were made to the registry and file system are collected. Depending on the software complexity and the hardware equipment, this process can take a few seconds or even an hour. The subsequent configuration phase is meant to customize the program and prepare a default configuration that applies to every user. This includes deactivating the automatic update feature and performing product registrations.

Once the configuration is done, the Sequencer again collects the changes, but this time the process is much quicker as the program configuration typically affects only a few registry keys and files.

The last step provides the modification of shortcuts and file type associations as well as the creation of `Feature Block 1` which is the first part of the application that is downloaded over the network when streaming is used [19]. Finally, all captured files and metadata are bundled into the virtual application file that can be deployed to the clients or placed on a server for being streamed.

Customizations after the sequencing process

When a project is opened for customization it is possible to change the behavior of the virtual registry and virtual file system. One can specify if a registry key in the virtual environment has to be merged with the native counterpart or if it overrides the local namespace at this area. The same applies to the virtual file system where a directory can be merged with the local file system or it can override the files completely to ensure that the application does not see any files from the native file system.

Furthermore, an application can be updated by opening the package in the Sequencer and letting it unpack the application to the App-V mount point drive. After that, the application can be used like if it was installed because the App-V minifilter will again redirect all file requests and registry keys to the virtual components. Thus, the application can be launched to start an integrated update mechanism manually if there is one. Otherwise, the setup routine can be executed to make the changes that are required for the software update. Once everything is finished the monitoring phase can be stopped which will provoke the Sequencer to save all modifications into the existing virtual application file. Afterwards, this file has to be copied to the server and will then be streamed to the clients with the updated application.

The Sequencer also offers the chance to compress a virtual application file so that the client has to stream less data over the network [36]. This setting is automatically enabled if the file is greater than 4 GB, otherwise the compression is optional. This is because it requires more computing time on the client to decompress the data. Certainly, the virtual application file has a limit of 4 GB so it is not possible to create applications that are greater than that limit [41].

3.4.2. Management Server

The core component of Microsoft's Application Virtualization Management System is the centralized Management Server. It serves the deployment of all virtual packages through streaming and offers the management of applications, licenses, and reporting [23].

The Management Server can only be installed on a Windows Server operating system and thus cannot be used in networks that run Linux on their servers [10]. As it is also responsible for authentication and authorization, the administrator has to set-up an Active Directory that handles the users and groups as well as the group policies. These are required to control access to applications and administrative functions.

The main protocol for streaming from the Management Server is the Real Time Streaming Protocol (RTSP) which was designed for media streaming servers. As it is not stateless like HTTP, the connection can be re-established if the client got disconnected [45]. This is very important because the streamed files oftentimes have several hundreds of megabytes and would have to be streamed again from the very beginning if the connection was interrupted. Thereby, the transfer time would be much longer and the user would have to wait a long time until the application starts.

Additionally, App-V supports the HTTP and SMB protocol for streaming, which can be utilized to build an infrastructure that does not have Microsoft Servers or cannot be used because of firewall policies or the like.

In order to make applications available to the user, the Management Server offers a way to let the App-V Client connect to the server and retrieve a list of applications that he is allowed to use. This mechanism is called the `Publishing Process` because it lets the App-V Client create all the shortcuts and file type associations that are published to the user by processing the list. Later on, when the program is launched, the client streams the application file from the server's content share that contains all available applications [34].

3.4.3. Web Service, Data Store, and Management Console

The `Data Store` is a Microsoft SQL Server which holds a list of all virtual applications that can be assigned to users. Thus, it also saves the assignment and permissions for each user and application. Furthermore, if the reporting feature is enabled, it stores the usage information that have been sent by the App-V Clients.

As App-V provides the ability to configure how many licenses of a program are available, it has to store this configuration in the database and must also keep track of occupied licenses. If the pool of licenses is empty, App-V takes care that no further instances of this application can be launched, until another client finishes its execution.

In Microsoft's infrastructure, the `Web Service` component uses the web server application called "Internet Information Services" (IIS) which is the interface for all read and write requests that have to go to the database. Thereto, it handles the communication between the Management Console and Management Server as well as the Data Store [23]. Furthermore, it can be used for streaming over HTTP(S) instead of using the Management Server and RTSP(S) [9]. This enables companies to connect a central office where the Management Server is hosted, with several branch offices that are using the App-V Client to stream applications over the Internet. Thus, no new infrastructure has to be created, as most firewalls are already configured for HTTP. This also makes the integration of roaming users very easy and enables them to stream applications to their home computer or even to their laptop while working external. Last but not least the troubleshooting is much easier as administrators know HTTP better than RTSP and are familiar with the configuration of a web server.

The `Management Console` is used to manage the App-V environment and offers to add or remove applications, assign them to users, and configure licenses.

3.4.4. Streaming Server

The streaming feature loads a virtual application from a server into the local cache of the client. This happens on demand when the application is started the first time and thus is not cached yet. Moreover, an application is streamed when the cached version is older than the one on the server.

As the size of a virtual application file can be several hundreds of megabytes or even a few gigabytes, the streaming process over the network can take a very long time, until every bit is on the client. This is where the two feature blocks come into play: after the application's launch `Feature Block 1` is streamed first because it only includes the necessary blocks a program needs to get started. Typically, the primary feature block is 20 to 40 % of the virtual application file [55]. Thus, the user can begin to use the software after a few seconds and does not have to wait for the whole package. Afterwards, the rest (also called `Feature Block 2`) of the application file is downloaded into the client's cache. Concerning, the block-wise storage and the RTSP protocol a client can request special blocks inside the virtual application file when the program requires files or registry keys that have not been transmitted yet.

The next time the application is launched, the App-V Client checks for a new version and updates the cache if necessary. Otherwise, the program is started directly from the cache [41].

As the Management Server also offers the streaming feature, a separate Streaming Server is optional. But it can be more effective to host a server nearer to the clients if the Management Server is not hosted inside the same network. Thus, a Streaming Server can act like a caching machine that mirrors all virtual applications. In order to always serve the newest packages the Streaming Server has to synchronize its content share with the Management Server.

3.4.5. Client

As App-V is not able to execute the virtual applications without an agent it is necessary to distribute the App-V Desktop Client by a deployment system. Therefor, the setup provides a silent installation with many parameters to customize the client's behavior [8]. This includes the settings for the infrastructure type (off-line as stand-alone or on-line with streaming), the cache size and configuration options for the streaming, like the protocol and the address of the server from which the applications are streamed.

The Client can be considered as a launcher which handles the request to run the desired application and build the necessary virtual environment. So there is no way to start a program directly by executing the virtual application file like an EXE file. Instead, the App-V Client (in the case the publishing method is used) or the MSI file (in case of the stand-alone architecture) create special shortcuts that execute the App-V launcher and tell it by a parameter which application it has to start. Listing 3.2 shows an example of how a shortcut target looks like for the Word component of Microsoft Office 2010.

```
sfttray.exe /LAUNCH ''Microsoft Office Word 2010''
```

Listing 3.2: Command for launching Microsoft Office Word 2010

The send-to menu items work just the same, as they are only normal shortcuts in a special folder. Also file type associations like *.doc are handled this way, where App-V, for example, has to launch Microsoft Office when the document is double-clicked. Therefor, the App-V Client is registered as a launcher for the filename extension and has to start the virtual application for this specific extension by creating the virtual environment and running the program in there. These changes have to be made in the operating system and thus cannot be isolated as they have to be available to the native system.

Of course, the name of an application must be unique in order to start a program this way. The Sequencer identifies every application automatically and generates a name for it which can be adjusted at sequencing time if necessary (for example if a name conflict occurs).

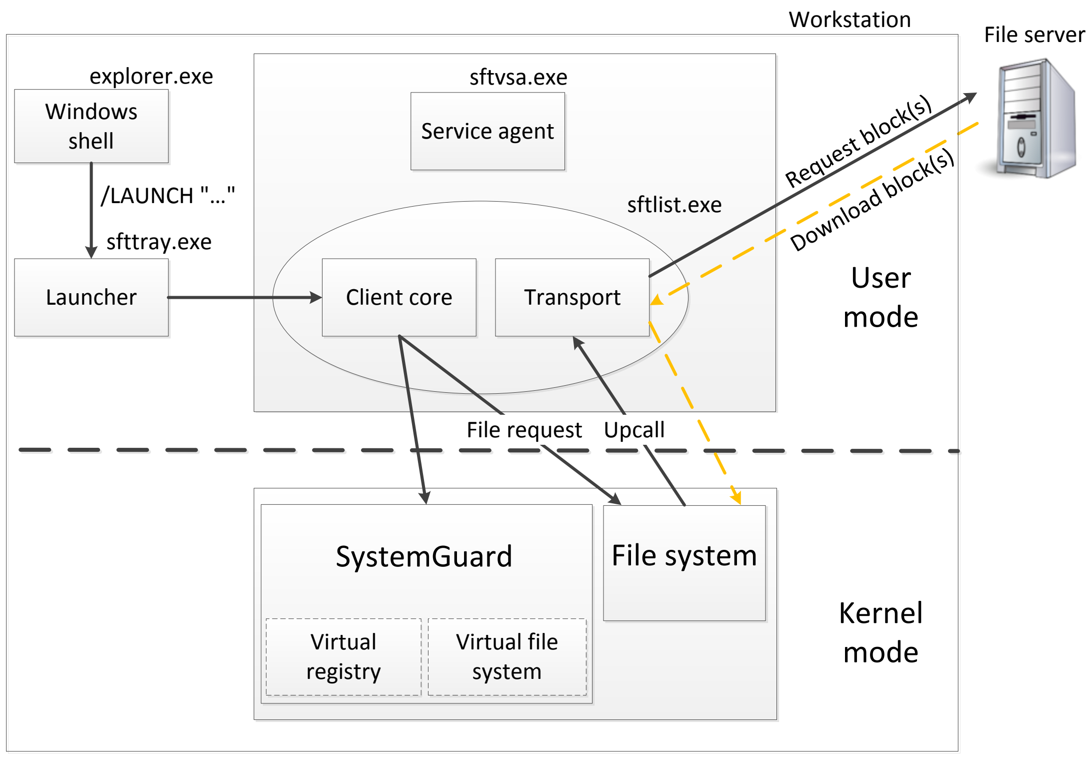

Figure 3.4.: Launching process of an application

The process of launching an application is shown in figure 3.4. If the user clicks on the application's shortcut item, the Windows shell respectively the process of Windows Explorer will start the App-V Launcher. This component decides with the help of the given parameter which application it has to start. It then creates a process for the App-V Listener which will send a request to the App-V volume file-system in order to get the required files that are needed to launch the application. If the application is not cached on disk and the streaming mechanism is used, the file system will send an upcall from kernel mode to the transport component that runs in user mode. This will fetch the requested blocks of the virtual application file from a server over the specified network protocol (RTSP, HTTP, or SMB) and hand it back to the file system [80]. After that, when the necessary blocks have been transmitted and are available in the file system, the Virtual Environment Manager will create the virtual environment. Then, the App-V Listener creates the application's process in user mode and hands it over to SystemGuard, which will transfer the process into the virtual environment. Even though the process is not created directly by the Windows shell as well as executed in the isolated environment, the application is represented in the process list with the same process name as if it would run directly on the system. SystemGuard will also ensure that every child process that is created will also be executed in the same environment. After the process has exited, the Virtual Environment Manager ensures that all configuration settings are written to the user's delta file in the roaming profile directory and destroys the environment [81].

3.4.6. App-V for Remote Desktop Services

The Remote Desktop Services Client provides the same functionality as the Desktop Client but is adjusted to run on terminal servers. These have to be hosts that have the Windows Server operating system installed, which means that it cannot be used with alternative operating systems like Linux or Solaris.

Here, too, App-V can solve common conflicts that appear with many programs, because an application often only works for the first user, for example, if it locks resources exclusively. This behavior is acceptable in single-user scenarios as only one person at a time can use the installed software, but on terminal servers with many users this is a knock-out criterion. Normally, conflicting applications have to be separated into different Remote Desktop Session Host servers which is very time-consuming and needs extensive tests. With App-V all conflicting applications are virtualized and thus can be located on the same server.

Thereto, the administrator installs the Remote Desktop Services Client on the terminal server to make it available for all users that start a remote desktop session. Then, the virtual applications can be used like any other program that is installed locally on the server [37].

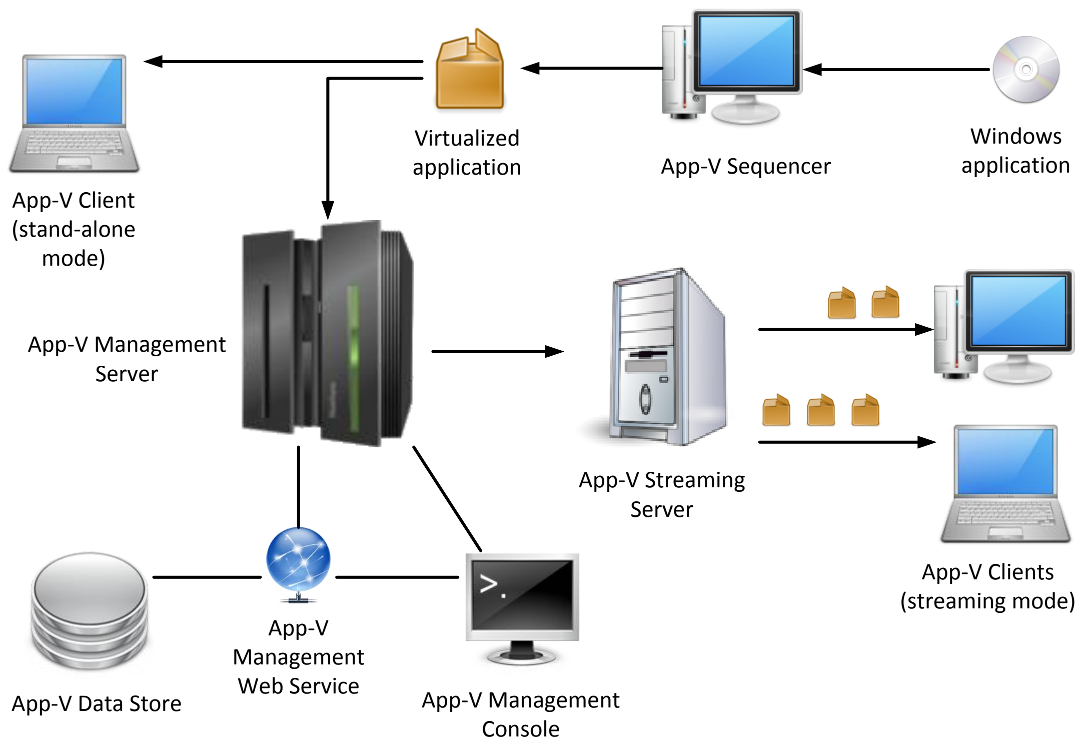

Figure 3.5.: Infrastructure of App-V

3.5. Architecture

The stations that an application passes on the way to the client are illustrated in figure 3.5. It also covers the full infrastructure with all components that are required therefor and the stand-alone mode as the opposite way of designing the environment without Microsoft's servers.

3.5.1. Full infrastructure

This type requires all components that were mentioned before. Therefor, the client needs to be configured to run in an on-line mode and talk to a server, whose job is to host the applications on a network share or a web server and provide a way for the client to stream the blocks over the RTSP(S), SMB or HTTP(S) protocol.

Publishing Refresh

In order to publish an application to the end-user, the App-V Client has to download a list from the server that contains all applications the user is allowed to use. The list is returned as an Extensible Markup Language (XML) file that contains several elements for every application. Including the path to the virtual application file, shortcut locations and file type associations.

The Publishing Refresh is the process of collecting the user-specific applications, building the XML file and establishing the application on the client. It is initiated at logon time or in an interval that periodically checks for updates to the list. When the client sends the request to download the list, the client computer has to be authenticated first by using the Kerberos protocol. Afterwards, the Management Server retrieves the applications that have been assigned to the user from the Data Store and builds the XML file. The App-V Client then evaluates the list and creates shortcuts as well as file type associations [34].

If the client was configured to cache all applications immediately after the publishing refresh instead of doing it at the first execution, the App-V Client will stream the virtual applications to the client and cache them locally.

Streaming

When a virtual application is executed by the user, the App-V Client checks if the locally cached version is up-to-date. In this case, the application will be started without any additional data transfer over the network. Otherwise, the new virtual application file will be streamed into the client's cache by first downloading the Feature Block 1.

Usage reporting and licensing

If activated, the client will send statistics on the application's usage to the Management Server as part of the publishing process. These reports include the name of the application and the user's account alongside the computer name and the time slot in which the application was in use. Such statistics can be helpful to manage the amount of available licenses. If the management discovers that too many licenses are unused, then money can be saved. On the other hand, if more users need to work with licensed programs than licenses are available, they can buy more.

3.5.2. Stand-alone mode

As a counterpart to the full infrastructure it is possible to run the client in two different stand-alone modes. Both scenarios require ways to create the necessary shortcuts and file type associations in a from the client decoupled manner. This can either be the MSI file that was generated by the Sequencer or a scripting language called SFTMIME. The latter offers ways to publish an application to an end-user or repair the application by resetting the user's delta file.

With streaming

Although the client operates in an on-line mode, this infrastructure type does not provide the publishing refresh and usage statistics, nor does it support licensing control. Therefore, shortcuts and file type associations cannot be generated by the App-V Client by fetching a list from the server. Instead, they have to be created by the application-specific MSI file or a SFTMIME script that is represented as a batch file and executed by the software deployment system. When using the MSI file it is necessary to specify a parameter at installation time in order to tell it that the streaming functionality is used and where the virtual application file is located. Thus, the App-V Client is able to decide which applications have to be streamed and which applications were directly cached. This enables the usage of streamed and only locally cached applications at the same time.

Without streaming

In this mode the client completely operates off-line and thus is not able to stream the applications from a server. This requires the MSI file or SFTMIME script to additionally load the virtual application into the cache as it would not be possible to execute the program otherwise. Therefor, the virtual application file has to be deployed to the client along with the MSI file or SFTMIME script which can be done by any type of software deployment system.

One disadvantage of this mode is that the cache might run out of capacity and thus applications have to be removed from the cache if a new one has to be imported. Thereto, the App-V Client uses a Least Recently Used (LRU) algorithm to decide which application will be removed [34]. Those programs will then be unusable until the MSI file is reinstalled or a SFTMIME command is used to cache a displaced application again. Thereby, the problem is only temporarily solved because then another application will be removed from the cache. When streaming is used, the cache can be refilled automatically if it is not present, but in the case without streaming App-V cannot determine the package location in order to cache it again.

3.6. Interaction between isolated applications

Obviously, one goal of application virtualization is to isolate programs from each other. But sometimes the separation of applications might not be desirable because they depend on each other. This is mostly the case when middleware or plug-ins are virtualized and have to be used by other virtual applications. By default it is impossible that, for example, a virtualized web browser uses a virtualized Java Runtime Environment that resides in another package because they do not share the same virtual environment while they are running.

To allow the interaction between virtual applications App-V provides a concept called

`Dynamic Suite Composition`. Therewith, a package can be defined as a primary application that uses several secondary packages as dependencies. So, for example, the globally unique identifier (GUID) that identifies the Java Runtime Environment (JRE) package will be used as a reference inside the Firefox package, which depicts the primary application. When Firefox is launched, App-V utilizes the dependencies of this package and executes them in the same virtual environment. This enables all applications that have been grouped into the same environment to communicate with each other and access files that belong to another package.

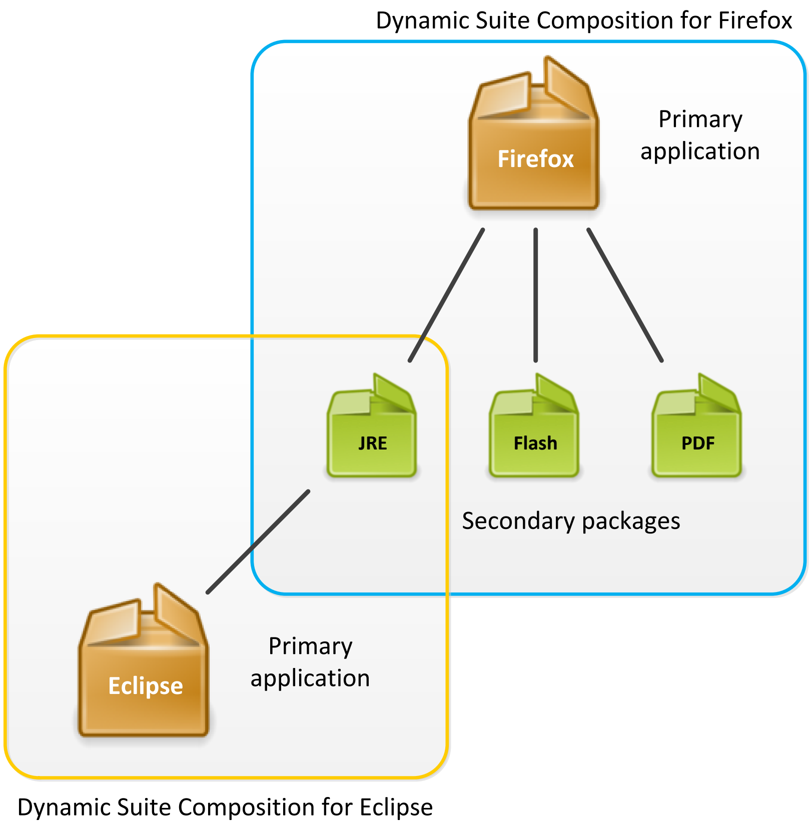

Figure 3.6.: Two independent Dynamic Suite Compositions

The concept of grouping packages to suites makes it possible to use an application as a dependency with many primary packages, which is illustrated in figure 3.6. Thus, it is not necessary to include a middleware like the Java Runtime Environment inside every package that depends on it. Hereby, the primary packages can be kept small because they only need to contain the main application while all the dependencies are sourced out into separate packages. This also makes it fairly easy to update a secondary package as it has to be done only once.

In addition, besides specifying the GUID it is possible to flag the referenced dependency as mandatory. This means, that if the streaming feature is used, the primary feature blocks of all packages, on which the primary application is depending on, must be present in the local cache. However, there might be situations where a secondary package cannot be loaded and thus fails to start. If it was defined as being mandatory for the primary application the program will also fail to launch [20].

3.7. Server Application Virtualization

To not only virtualize client programs but also server applications, Microsoft is currently working on a new product called Server App-V. At this time it is available as a release candidate and will be included in the System Center Virtual Machine Manager (SCVMM) 2012 [86]. So it will not be part of the App-V software bundle which is part of the Microsoft Desktop Optimization Pack (MDOP).

Server App-V has the same goal as the Desktop Client. It creates applications abstracted from the operating system that cannot cause conflicts and are easily portable to other servers. The main difference to the Desktop Client is that the server variant allows virtual applications to communicate with natively installed ones. It also integrates virtual applications seamlessly into the operating system if necessary, which is not supported by the desktop version yet. Thus, Windows Services are registered as if they were installed locally, but they are still virtualized. Also the interprocess communication like COM, DCOM, and COM+ is supported which is an essential component for server applications that need to exchange data [62].

4. Integrating App-V into a heterogeneous network

In the year 2000 Munich's department of education started to equip the public schools and urban nurseries with a local network, a server and workstations. Thereto, a new bureau was founded that today is called "Zentrum für Informationstechnologie im Bildungsbereich" (ZIB) [47]. All these school networks are connected and are centrally managed by this IT department. The goal of this project was to build a consistent environment which allows teachers and students the use of computers and educational software in class.

It is obvious that applications and users at some point require better hardware and software. Also the support for an operating system ends at some time and thus no more updates are available from this point on, which causes high security risks if an upgrade is delayed. These are the main reasons why the system needs adaptations over and over again. This is why the Windows clients had to be migrated from Windows 2000 to Windows XP recently [64]. At this, not only the 28,000 clients that have to be migrated are a challenge, also the over 700 applications need to be adjusted, tested and deployed [46]. Therefor, ZIB holds amongst others a partnership with T-Systems whose developers support the currently deployed system as well as working on its improvement by designing the next version based on the specifications of ZIB.

To make things easier with the upcoming migration to Windows 7 and to enable schools the usage of different versions of the same product without running into conflicts, one part of the migration plan is to integrate an application virtualization strategy.

4.1. Infrastructure of Munich's school network

The network as a whole consists of about 400 school servers [46] that are hosted decentralized and have SUSE Linux as an operating system installed. Every campus, especially if it is a school, has at least one server and a few hundred workstations that are mostly running Windows XP at the moment. Some schools (like vocational schools) have special needs in software and thus already have Windows 7 installed on a few unmanaged machines.

All deployed software is either packaged as a MSI file or by a proprietary packaging product. Both are processed by an enterprise software deployment product that installs, updates and removes programs as well as drivers. Teachers with special rights are allowed to assign software to users or workstations which gives them the freedom to define different software sets for every user (pupils as well as teachers). For the administration of the network this freedom makes it hard to ensure that no conflicts arise with the installation of software because the potential combinations of applications cannot be tested anymore. Here, the virtualization of applications would be a great benefit for the maintenance of workstations.

4.2. Reverse engineering the infrastructure of Microsoft App-V

App-V provides the integration into networks that do not have the full infrastructure with all Microsoft Windows Servers. But the documentation does not contain a complete strategy on how to replace their components with alternative ones. So, for example, Microsoft does not offer any information about the authentication process between the App-V Client and Management Server. They claim, this is only possible when using their full infrastructure, but there is a way to get the client authenticated even without a Windows server.

This section will explain how the documented features are integrated into an existing network that does not use Windows servers and how some of the undocumented features can also be achieved.

4.2.1. Streaming and authorization

As the school servers do not have a Windows Server operating system installed, these cannot be used as Streaming Servers as suggested by Microsoft. Thus, to get the streaming feature working with a Linux server, App-V offers the HTTP and Server Message Block (SMB) protocol. The first requires a web server for Linux, like the Apache HTTP Server, while the latter requires a file service like Samba.

Using the HTTP protocol offers the opportunity to stream applications outside the network over the Internet. As this is a potential security issue, the secure HTTP(S) protocol should be used to ensure the client only streams applications from a trusted origin. Therefor, a certificate has to be established between the web server and the App-V Client. The documentation of Microsoft describes how a Management Server or the Internet Information Services (IIS) software has to be configured in order to use a secure communication [24]. But there is no official documentation or even a tutorial on the Internet, on how to set-up an alternative web server for (secure) streaming. Furthermore, the IIS offers the Integrated Windows Authentication which ensures that only authorized users gain access to the applications and that they only can stream applications that have been assigned to them.

As the configuration of the web server requires more effort than the file service, I chose to research the SMB protocol for streaming in more depth. This variant is easier to implement because the authorization can be controlled by access control lists (ACLs). Thereto, the standard ACLs of Linux are used by creating an individual group for every virtual application. Then, the group has to be assigned to the directory that contains the application and the file system permissions have to be set to `read` and `execute`. Thereby, every user that has the right to use the application needs to be assigned to the application group. The App-V Client then needs to be configured to stream the packages from a specific network share that is located on the file server. As the Samba file service is mostly used as a Primary Domain Controller (PDC), it will authenticate the user when he logs in on the workstation or if he accesses a network share. This is an easy way to protect the application on the server and to make sure that only authorized people can access it. The PDC currently acts as a replacement for the Active Directory which would require a Windows Server operating system on the school servers.

4.2.2. Publishing and authentication

The only alternative to implement the publishing feature without a Management Server is to use a web server and the HTTP(S) protocol. Thereto, the App-V Client is configured to get the list of applications from a specific Uniform Resource Locator (URL). Beside the before mentioned security risk of HTTP and missing authentication, there is also a technical downside when using a web server that is not the IIS. As the URL is the same on every App-V Client and does not contain any user-related information, there is no way to generate a user-specific XML file with the applications he is allowed to use. So on the server-side each request is anonymous and thus allows only one application list for all users in the network.

The documentation of App-V stated that the Management Server uses the Integrated Windows Authentication mechanism to authenticate the Client. This functionality is also offered by the Apache web server through a module and allows to authenticate an user over a website by using the workstation logon credentials instead of prompting the user to provide his user name and password. A test with the Apache2::AuthenNTLM module[1] showed that the App-V Client can be authenticated against Apache in the same way, which solved two problems:

1. The list of applications is protected and only accessible by users that are authorized.
2. After a successful authentication the web server opens a session and saves some user-related information, including the user name. This makes it very easy to write a script in PHP or Perl that reads the user name from the session and generates a dynamic application list by collecting the data from a database and building the XML file out of it.

As the current system already saves user information, group memberships, applications and assignments of users to applications in a directory service that is accessed with the Lightweight Directory Access Protocol (LDAP), the integration of a script that generates the XML does not require big changes to the existing configuration. It only needs a way to differentiate between virtual applications that come into consideration for the XML and those application that are not virtualized and must be installed in the traditional way.

[1]`http://search.cpan.org/~speeves/Apache2-AuthenNTLM-0.02/AuthenNTLM.pm`

4.2.3. Reporting

The reporting feature can be enabled either by changing a registry setting for the App-V Client or by adjusting the publishing XML file which provides an attribute for this configuration.

Normally, the Management Server would collect the reporting data that is sent by the client. To clone this feature with an alternative web server it requires some script that accepts the HTTP POST request and processes the incoming data by parsing the XML and storing the information in a database. As the POST request happens directly after the Publishing Refresh GET request, it has to be the same script that handles both consecutively.

4.3. App-V as part of the Windows 7 migration project

As already mentioned, the application virtualization has to solve conflicts between the installed software. This are programs like Adobe Creative Suite in version 2 and 5 or Office 2000 and 2010. They have to be installed side by side because of educational reasons. As the programs install dynamic link libraries in different versions it is impossible to use them at the same time when they are installed natively.

With App-V this problem should be solved primarily. But it, furthermore, makes it easy to port the applications from one operating system to the next higher version without the need to sequence them again. This is possible because the virtualization layer overtakes the job to adopt the virtual application's system calls, file system paths and registry keys to the native ones provided by the Windows library. When there are required changes for the application to meet the new library version, Microsoft releases a new version of App-V with an adapted virtualization layer. The administrators only have to deploy the new version of App-V to the workstations without having to re-sequence the packages. When the MSI deployment is used, it is necessary to edit the packages and adjust the supported operating systems to the newer version. This is a simple task and it is by far not as time consuming as it would be to sequence all programs from the scratch.

But it is still necessary to test the packages for compatibility with the new version of Windows because App-V does not solve incompatibility issues of the software with the operating system. This is a very important thing to mention as some might think this problem can be solved with App-V, too. For example, it is not possible to get QuickTime version 3 running on Windows 7 64-bit because the setup refuses the installation on a non 32-bit architecture. Also registering the DLLs manually does not work because they were compiled to an old version of the Windows library.

Furthermore, applications like Autodesk AutoCAD which are complex and install a lot of libraries in the system folder are intended to be virtualized with App-V, too, because the software packaging process for such programs is very hard and time-consuming.

In order to find out if App-V can accomplish all those needs and to test its feasibility, it is part of the Windows 7 pilot experiment that will be carried out on a few schools. This test run is used to let some pupils and teachers test the current state of the new system and to find out the things that are not working as expected yet.

Based on my research on how to integrate App-V into a heterogeneous network, the integration into the school network can be summarized like shown in figure 4.1.

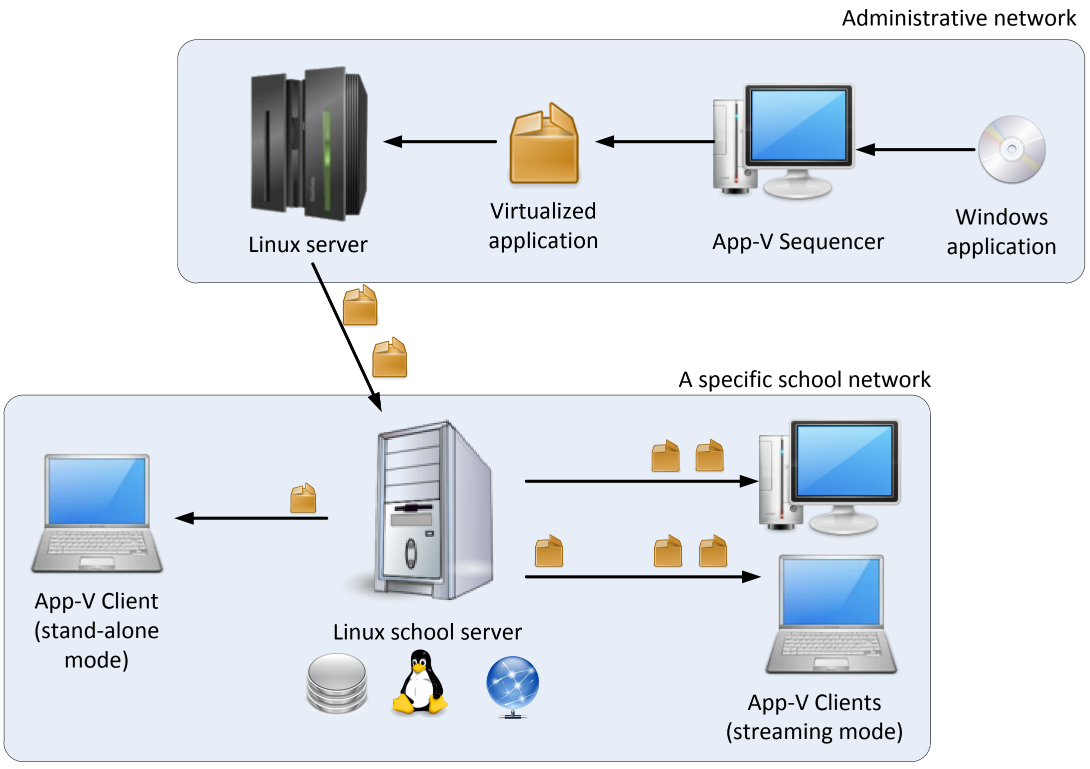

Figure 4.1.: App-V integrated into the school network

Like in the full infrastructure of App-V, the application is virtualized with the help of the App-V Sequencer. After that, the virtual package will be placed on a special Linux server instead of the App-V Management Server. This Linux server is responsible for the deployment of all software packages to the school servers, including the copying of the virtual application file and MSI file as well as the creation of a record for this application in the LDAP directory service. The deployment process does not have to be adjusted for the integration of App-V, but it might be of interest to extend the entry for applications in the directory service with a new attribute to differentiate between virtual applications and others. But this is not required if the stand-alone mode (with or without streaming) is used as they do not support the Publishing Refresh.

After the files have been copied and the application is registered in the system, the new application can be assigned to users and machines. Depending on the type of distribution, the software deployment system or the App-V Client will then publish the shortcuts to the users.

4.3.1. App-V Client

The client software is installed on all workstation computers and notebooks by the existing software deployment product. As App-V has to proof in the field if it can fulfill the needs, the stand-alone mode has been chosen to be implemented at first. This is because it requires no special changes to the existing server configuration and integrates with the least effort. Thus, the App-V Client's network setting is set to "offline" at installation time.

Certainly, the stand-alone mode can be changed very easily later on if the decision is to use streaming or even the publishing method. Therefor, the registry settings for the App-V Client must be adjusted accordingly and the configuration on the server-side has to be fulfilled in order to support streaming.

To keep the likelihood of a full cache small, the App-V Client is configured to increase the cache size as long as the remaining free disk space is at least 5 GB. As mentioned in chapter 3, the least recently used applications are displaced if a new application is cached and if the free space is not enough. This is, what the support team and help desk must be aware of if a user reports a problem about applications that cannot be launched. The first thing they always have to do in order to solve this problem is to reduce the number of assigned virtual applications. Secondly, to fill the cache again, the application has to be removed from the system and then installed again. A new installation in this case means that the MSI file has to be installed again to initiate the caching.

Alternatively, the problem could be solved by a script that queries the LDAP directory service and executes SFTMIME commands to cache the applications again which are currently assigned to the user and machine. The advantage of this approach is that the user does not have to log off from the workstation and go through the process of a reinstallation afterwards which saves him time. As the manually execution of a script is not user-friendly this approach might be too complicated for teachers and pupils but is feasible for the support team.

The same process could also be initiated through the web interface which provides the assignment of applications, too. Thereby, the web interface could offer an option to repair the cache on a specific workstation by clicking a button. This action would result in a command on the server that establishes a connection to a special background service on the client that waits for instructions from the server. Then, the server could send the SFTMIME commands which are executed by the client. Therefor, the already existing background service on the client and the web interface would require a few adaptations.

4.3.2. Mobile users

The school network also includes notebooks that can be used as a network-connected or off-line driven machine. These are treated like the always connected workstation computers with the difference that they allow the user logon without having a connection to the server, so everyone is able to use the installed software even in off-line mode. In order that the logon works without contacting the server, the user has to do the login procedure once while the notebook is connected, to cache the user profile and login credentials.

When application virtualization is used it must still be possible to use the programs in off-line mode. With the stand-alone mode of App-V there is nothing special to consider for notebooks, as the virtual packages are deployed like the other programs.

The situation is different when the streaming functionality is used as all the necessary applications have to be cached locally before the notebook is taken off-line. Otherwise, the user will not be able to use the programs.

Thereto, is is possible to set the cache behavior at the installation time of App-V. Per default, only applications are streamed to the client's cache that have been recently used. This configuration is good for workstations but not feasible for notebooks as the applications have to be available when the user starts to use them in off-line mode. One solution is to let the user launch all applications that he is going to need which however is very time-consuming and not user-friendly. The far better way is to change the mentioned cache setting to AUTO LOADING which means that all applications are streamed and cached locally right after the user logs on to the machine. So afterwards every program can be started when the laptop is disconnected from the network.

In order to have different versions of the App-V Client for desktop clients and notebooks, it requires to deploy two different software packages that include the necessary parameters for each configuration.

4.4. Software

Table 4.1 lists all applications that will be part of the pilot experiment momentarily and thus I researched them to analyze if they can be virtualized. The IT department will decide which of the successfully virtualized applications will be deployed to the workstations to test App-V and which are going to be installed natively. A "✓" indicates a successful virtualization while a "–" means that the application could not be virtualized. The sign "(✓)" denotes that the application could be virtualized but with limitations.

This section will outline several pitfalls that arise with the virtualization of software, especially with the commitment of App-V. For that purpose I will cover different software products of table 4.1 that will exemplify the common problems. Additionally, this section will present solutions on how to circumvent the problems and drawbacks, where possible.

4.4.1. Adobe Reader

Virtualizing Adobe Reader means that all other software that depends on it, cannot use it anymore because they cannot access the application inside the virtual environment. Thus, the feature of viewing a PDF embedded in the browser will stop to work. This can be circumvented by also virtualizing all the programs that interact with Adobe Reader, for example, Mozilla Firefox or Microsoft Internet Explorer, and then defining dependencies

Application	Virtualized	Comments
7-Zip	(✓)	No context menu integration
ActiveState ActivePerl[1]	–	Required by several programs that cannot be virtualized
Adobe Creative Suite	✓	
Adobe Flash Player	✓	Requires a virtual browser
Adobe Shockwave Player	✓	Requires a virtual browser
Adobe Reader	(✓)	Requires a virtual browser in order to use the plug-in
Apple Quicktime	✓	Requires a virtual browser in order to use the plug-in
AutoCAD Design Academy	✓	
FileZilla	✓	
GIMP	✓	
Hardcopy[2]	–	No interaction with natively installed applications possible
Hot Potatoes[3]	✓	
iTALC[4]	–	Uses a boot-time service and needs local interaction
IT Hit Map WebDAV Drive	–	Uses a boot-time service
IrfanView	✓	
Java Runtime Environment	✓	
Kaspersky Anti-Virus	–	Antivirus software must not be virtualized
LibreOffice	✓	
Microsoft .NET Framework	–	Required by the App-V Client, must be installed natively
Microsoft Office 2010	✓	Requires a Deployment Kit for App-V and an extensive documentation to get it virtualized
Microsoft Visual C++ Runtime	–	Required by the App-V Client
Mozilla Firefox	(✓)	No default browser and Jump List
Mozilla Thunderbird	(✓)	No default client and Jump List
PDFCreator	–	Printer drivers must be installed natively
Phase 5[5]	✓	
RealNetworks RealPlayer	✓	
VLC Player	✓	
WS_FTP LE	✓	
WinSCP	✓	

[1] Perl language distribution for Windows
[2] Screenshot tool
[3] Creates exercises
[4] Remote desktop control based on VNC (Virtual Network Computing)
[5] HTML editor

Table 4.1.: List of software that is part of the Windows 7 pilot experiment

between the Reader and any other secondary package. As mentioned earlier, it is not supported by Microsoft to virtualize the Internet Explorer, so it is necessary to just launch it inside the virtual environment of the Adobe Reader package. Thereby the natively installed browser shares the same namespace and has access to the isolated application. This requires to change all shortcuts on the system and the various ways of launching the Internet Explorer in order to have it executed by the App-V Client and placed inside a particular virtual environment. So, instead of calling iexplore.exe directly, the program invocation needs to be handled by the App-V Listener Service. For that purpose the adjusted browser shortcut has to be placed into the Adobe Reader package during sequencing [73].

For this reason it is not recommended to virtualize such a standard system component that may be used as a plug-in by several other applications, especially those that are like the Internet Explorer.

Nonetheless, this software can be virtualized and integrated into the system as the standard viewer for PDF documents. However, to get the program running inside the virtual environment, one has to manually set two registry keys to disable updates and a security feature called the “protected mode”. To also remove the update service that would otherwise be running in the background while the virtual environment exists, one needs to edit the package in the sequencer and delete the service.

The deactivation of an update service is very important because a program update modifies the current state of the application. This means that files and registry keys are altered which are captured inside the user’s delta file because the golden image must not be changed. Thereby, the update grows the user’s profile and implicates that the user may encounter problems with this unmaintained state.

4.4.2. Adobe Creative Suite and Adobe Acrobat

When virtualizing Adobe Acrobat one has to keep in mind that it installs a printer driver which will not work inside the virtual environment. Thus, it is necessary to separate the PDF printer from the installation and deploy the driver natively.

The reason for virtualizing a complex bundle such as the Adobe Creative Suite is that some schools require to use two different versions of it in parallel on the same machine. Without virtualization this task is impossible to complete as the software installs many libraries that are incompatible to each other.

Certainly, a few problems arose that made the sequencing very time-consuming and complex. One issue was that the Help Center could not be started without local administration rights. Normally, when an application needs higher rights, the Windows 7 User Account Control (UAC) takes control and elevates the user’s privileges. In this case it even did not work when the virtual application was launched as an administrator. The solution to this problem was to edit the virtual package and set an environment variable to tell the application to execute without any demand [40].

Adobe Bridge and Adobe Illustrator both caused an error message at launch time. The first could not load start scripts from the application data folder in the virtual file system. Like all folders in the application data profile the start script files are user-specific and thus will be recreated by Adobe Bridge at runtime. So the solution was to exclude the start scripts from being captured in the monitoring phase respectively deleting them afterwards from the virtual file system.

The problem that caused the Illustrator's error message was not that obvious as it claimed that the machine does not have enough memory. Luckily, a thread in the Adobe software board covered this issue, which was not a specific virtualization problem. After following the recommendations of deleting a preferences file in the application data folder, the problem was solved.

4.4.3. Hardcopy

Hardcopy is a tool for taking screenshots and therefor it extends the Windows-integrated command for capturing the screen and saving it to the clipboard with several more features. Including a possibility to directly send the screenshot to a printer and adding an icon into the active window's titlebar for taking a screenshot with one mouse click.

So virtualizing this program with all its functions is impossible, because Hardcopy can only integrate the icon into windows that are also virtualized and share the same virtual environment. As this is typically not the case, this function will be lost because of the application's isolation and separation from the natively installed programs. Furthermore, capturing the keyboard shortcut for the print screen feature will only work when changing the LOCAL INTERACTION setting of the virtual application to "true". Otherwise, the App-V Client will not pass the command to the program while it is listening in the background.

For this reason we decided to install Hardcopy natively because it needs a better interaction with the operating system and other programs as App-V admits.

4.4.4. Microsoft Office

Like almost all Microsoft products the Office suite is well supported by App-V, but the effort of virtualizing them is very dissimilar between different versions.

The virtualization of Office 2000 is fairly easy as it does not install any license service like the newer versions. Thus, the sequencing was straightforward and encountered no problems.

In contrast, Office 2010 is much more complicated as it requires a few preparations before the sequencing phase can begin. In addition, to get the application running virtualized on the end-user's computer, it is necessary to install a middleware software called Microsoft Office 2010 Deployment Kit for App-V. The distribution can be achieved the same way as the App-V Client is installed on the machine. The Deployment Kit allows the integration of the virtualized Office into the system and enables interaction between the suite and other applications.

Following Microsoft's documentation on how to prepare the machine setup and installing Office 2010 it is possible to sequence the application successfully [42]. That said, it would not work out if Microsoft did not support virtualizing their big products, because they are too complex and could not be sequenced without assistance. Furthermore, a virtualized Office would miss features like the integration of the Windows Desktop search if the Deployment Kit was not there.

4.4.5. Mozilla Firefox and Thunderbird

I thought that Firefox and Thunderbird will be good programs to become familiar with sequencing because they have no dependencies or known components that would cause huge conflicts in the system. As there are also existing unofficial portable versions of these applications, my presumption was strengthened.

However, these programs already raised a few points that needed a deeper exploration of App-V's functionality as I might have expected.

Setting a program as default

Normally, at installation time, Firefox can be set as the default browser while Thunderbird can be the preferred mail client. With App-V these settings cannot be applied to the operating system, even though the Sequencer recognizes them. This is because neither the publishing process nor the MSI file currently support the integration of an application as default. In the stand-alone infrastructure this drawback can be circumvented by adjusting the MSI file with an editor like Wise Package Studio[2] and adding the functionality this way. Alternatively, as this setting consists of adding or changing keys in the registry, this modification can be accomplished by the software deployment tool as well if it supports the modification of the registry.

Application profile

The two programs are using locations in the user's Windows profile, where user-specific data like extensions, configuration settings, bookmarks or mails are stored. At sequencing time it is important to known if the application profile should be a part of the virtual environment or if it can be saved into the default location in the native file system.

The fundamental difference between these two options is that including the profile into the virtual environment prevents other (locally or virtualized) applications from accessing it. Sometimes it could be desirable to use different versions of the Mozilla software with

[2]`http://www.symantec.com/business/package-studio`

the same profile. Thus, the application profile has to reside in the user's profile and must be excluded from the virtual package.

Certainly, this has the drawback that there is no possibility to create a default profile for the virtual application inside the package that presets a configuration for all users. This, for example, is necessary to provide a preset home page, to customize the program for the school network and to disable the update feature. However, as Firefox and Thunderbird can be configured to use a centralized stored configuration file on the server [71], there is no need to prepare a default profile. This auto-configuration is already implemented into the school network and can be used for a virtual as well as for a natively installed application.

Plug-ins

As mentioned before, Firefox does not have special software requirements or dependencies, but it can use plug-ins of other applications to play embedded media or display PDF documents inside the browser. This requires the local interaction of the virtualized Firefox package and the software that provides the plug-in.

Alternatively, such plug-ins can also be virtualized and defined as dependencies of the Firefox package, which enables all packages of the Dynamic Suite Composition to share the same environment and see each other.

Jump Lists

Virtualizing Mozilla's web browser and mail client showed up another program-related limitation of App-V: the integration of a virtual application's Jump List into the taskbar. Such Jump Lists were newly introduced by Windows 7 and can be seen as special context menus that are used individually by every program that offers support for this feature.

Firefox not only offers shortcuts to the most common tasks like opening a new tab or window. Its Jump List also displays the most frequently visited pages to make them quickly accessible for the user.

All entries in the Jump List that link to the program's executable are not working because the shortcut points to the virtual drive of App-V. As applications have access to their own package folder only inside the virtual environment, clicking on the shortcut results in an access restriction because the shortcut is integrated into the taskbar outside the virtual environment and thus the user is not able to run the program from the protected virtual drive.

The same applies to Thunderbird, but it is not a general problem of App-V because the Jump List is working in other virtualized products like Office 2010 or Adobe Reader.

It is likely that the Mozilla Foundation implemented the Jump List in a way that App-V is not aware of. Normally, the App-V Client would modify the shortcut targets and replace them with the App-V launcher executable, like it does to the shortcuts on the desktop or in the program menu.

4.4.6. Investigation of problems inside the virtual environment

Sometimes it might be of interest to have a look inside the virtual environment in order to investigate problems of a virtual application after the sequencing. As App-V restricts the access to the virtual drive, there is no way to browse the file system with the Windows Explorer. Instead, an alternative file manager must be used and opened inside the virtual environment. Therewith, it is possible to observe the files inside the package directory and even those that are redirected by the virtualization layer are discoverable in the system folders.

On the other side, to have a look into the virtual registry, the registry editor can be launched inside the virtual environment. Therefor, App-V provides a special command like shown in listing 4.1. In this example, the App-V Launcher (sfttray.exe) creates the virtual environment for Microsoft Word and executes the registry editor (regedit.exe) inside of it. Thus, the editor will present all registry keys of the native namespace as well as the ones that are only available inside the virtual environment of Word.

```
sfttray.exe /EXE regedit.exe "Microsoft Office Word 2010"
```

Listing 4.1: Command for launching the registry editor inside the virtual environment

Instead of the registry editor executable any other program can be used as a parameter as long as it is not the Windows Explorer (explorer.exe). This can be useful in order to find out file conflicts and access restrictions that may occur inside the virtual environment. Thereto, the Process Monitor of Windows Sysinternals is a very helpful tool because it shows the real-time file system activity as well as requests to the registry and it allows to monitor processes and threads [25]. In order to have the virtual application monitored by the Process Monitor, it has to be launched like shown above because it would otherwise capture the redirected requests.

One solution for a file conflict could be to exclude the file from the environment if there is a way to provide it outside the virtual application. This, for example, might be the case for application profiles that can be placed into the user profile instead.

4.4.7. Documentation of proceedings and troubleshooting

In order to document the proceedings that are necessary for the virtualization of each application I installed a wiki for that purpose. Therefor, I chose the MediaWiki software application because it is highly customizable and still easy to use. Furthermore, as Wikipedia also uses this wiki it will be enhanced and supported over a long period of time.

This wiki not only includes the information for App-V applications but also the specifications for the installation and configuration of the applications that have been made by the

IT department. An employee of T-System also contributed to this list as he was responsible for creating all of the applications as unvirtualized packages. In cooperation we collected the parameters for the silent installation of the programs and created both an unvirtualized application and an App-V package where possible.

Additionally, I documented the pitfalls and troubleshooting of problems that occurred while sequencing the applications. This not only was helpful for myself because I could check up which approaches to a problem I already had tried. First and foremost the documentation shall help the ones that adopt the sequencing of App-V in the future. Thereby, they do not have to solve the same problems again and they will have an easier start with virtualization because they have several examples of packages that are ready.

Moreover, I wrote a documentation (see appendix A) about the App-V development environment, how App-V works and how it is integrated into the school network. It also describes the sequencing process as well as the management of existing packages, including software updates and modifications. This document is intended to be a manual for those that do the network administration as well as for the software packaging engineers which will create applications with App-V.

4.5. Categorization of applications

This section will give an overview about which programs are good candidates for application virtualization and which are not.

4.5.1. Not virtualizable

Some software should not be virtualized or is technically not supported to be virtualized. Including Windows updates and operating system service packs which would have no effect if they were isolated from the operating system. Additionally, it would cause a security problem to virtualize those components. Furthermore, operating system components like the Windows Update Service or the Windows Firewall may not be virtualized because of security reasons. The same applies to third-party security software like client firewalls, antivirus software, and encryption software. All of these must have direct access to the operating system and its components, including the file system and registry. If they were virtualized, the redirection of files and registry keys would separate them too much from the system.

Most of the aforementioned could even be virtualized with kernel mode agents only, because such software needs to run with higher privileges in order to have access to all parts of the system.

Moreover, device drivers are not supported to be virtualized because of their integration into the operating system. In addition, applications could not communicate with isolated drivers and thus it only makes sense to install them locally. Besides that, if the virtualization layer fails the driver would not behave the way it should and might cause a system crash.

Also the Internet Explorer can be considered as an operating system component that is highly integrated into Windows. Although Microsoft supports the virtualization of the Internet Explorer with terminal services, the Windows XP Mode, and MED-V[3], they do not support it with application virtualization [39]. It even violates the end-user licensing agreement when multiple versions of Internet Explorer are running on the same instance of Windows [27].

Applications that are dependencies for other programs might not be virtualized if one of those primary programs cannot be virtualized. This is because a locally installed program does not have access to a virtualized application and thus cannot use it. If all programs can be placed into the same virtual environment without any reservations, then there is nothing to be said against it. Otherwise, all applications should be installed natively. For example, the Microsoft .NET Framework is used by many applications and thus should not be virtualized. Most notably, the framework is required by the App-V Client and must be installed locally. Certainly, the .NET Framework allows the usage of different versions side by side which renders its virtualization unnecessary [30]. Another example is the Java Runtime Environment which cannot be used in a command-line interface (CLI) like the cmd.exe or the Windows PowerShell if it is virtualized. This is because the CLI is not able to access the binaries like the Java compiler (javac) outside the virtual environment.

The interprocess communication between applications is also not possible when they are not running in the same virtual environment. Those programs must also be installed natively.

4.5.2. Virtualizable, with reservations

Services are mostly supported inside virtual environments but do have some limitations when it comes to services that need to be started at boot time. This cannot be achieved in the traditional way because isolated services are not registered as the ones that are locally installed. Thus, it is not possible to let them automatically start or define them as dependencies for other services. The first drawback can be circumvented but an identical behavior will not be achieved.

Furthermore, applications that install drivers have to be separated from those components in order to virtualize the application on its own and install the driver natively. This, for example, might be a printer driver like Adobe PDF (formerly known as Adobe Distiller) which creates PDF files. Certainly, the separation of such parts is not always an easy task and will take some time. It also needs more testing of the virtual application and the natively installed component in order to find out if they play together without problems. In addition, the decoupled component can cause conflicts that need to be addressed without the help of application virtualization.

[3]Microsoft Enterprise Desktop Virtualization, a centrally managed desktop virtualization solution

Moreover, applications that require special hardware like software protection dongles or a specific media access control address for licensing can be virtualized with App-V but they are bound to a certain hardware or workstation [1]. This limits the abstraction and reusability of a virtualized package but it sticks to the licensing agreements that a software brings along.

Also the integration of a virtualized application into the operating system is not fully supported by App-V yet. For example, the registration as the default browser or the integration into the context menu does not work. The latter is not supported because the Windows Explorer would require access to the virtual environment [87]. This cuts some features of an application which the user would expect to function and behave as he is familiar with.

Additionally, applications with a high utilization of the disk could be candidates for this category because the redirection that has to be made for a file handle at every request slows down the working. This, for example, is the case for database applications which would not be virtualized on a server then [48]. However, on a desktop with modern hardware it should be feasible to run the Microsoft SQL Server virtualized. Admittedly, for servers and terminal services the Server Application Virtualization (see section 3.7) will be a much better solution.

Further, applications that use hard-coded paths can cause problems, as it was seen in the Firefox example. This one only resulted in an unusable Jump List, but there are other programs that use configuration files with pre-defined locations or access files in the program code by using hard-coded paths which then might not work as expected. Those programs have to be recognized and must be installed to the default location instead of the virtual drive Q.

4.5.3. Virtualizable

There are programs, especially middleware like the Adobe Flash Player, that can only be installed to the system drive. As mentioned in chapter 3, Microsoft proposes to install software to the virtual drive Q to gain the best performance. However, the difference between an installation to the Q drive and the standard location is mostly not noticeable. Especially not with middleware software that includes a small number of files that need to be redirected by the virtualization engine. It is more drastic when complex programs like Microsoft Office or Adobe Creative Suite are accessing most of their files from the system drive because those requests have to be intercepted and redirected to the virtual drive.

5. App-V compared to other products

This chapter will introduce a few products of other competitors in the application virtualization market and explain how they differ from Microsoft App-V. As the main focus of this thesis is App-V, the following comparison is based on whitepapers and product reviews without having them tested on my own.

5.1. VMware ThinApp

Like Microsoft did with SoftGrid, in 2008 VMware acquired the company Jitit Inc. that developed an application virtualization product called Thinstall [59]. Afterwards they renamed it to ThinApp, that is – contrary to Microsoft App-V – also available as a trial version.

The fundamental difference to App-V is that ThinApp is an agentless virtualization product, which means that it does not require any client software or requirements being installed on the desktop machine to run the virtual applications. Thus, all applications are bundled to EXE files that include everything the program needs to run. However, this requires ThinApp to encapsulate an agent-like Virtual Operating System (VOS) into every virtual application for building the environment and starting the program inside of it. Certainly, ThinApp executes its VOS and the application in user mode which does not require administrative permissions for the user to run an application nor can a crash affect the operating system [58]. A drawback of this agentless approach is that every package must be changed if VMware releases a new version of ThinApp in order to be able to use the new features.

ThinApp also offers a streaming functionality but does not cache the applications locally on disk. Instead, only the necessary parts of the executed application are loaded blockwise from a server or any other network drive into the memory. Alternatively, the portable executables can be distributed by a software deployment system or even on a data storage device like a USB flash drive. This is possible because it does not require any rights to run the program or install a MSI file like with App-V. Thus, an application can also be executed by external users that do not have anything special like an agent installed on their notebook.

As a counterpart to App-V's Publishing Refresh, ThinApp uses login scripts to publish the shortcuts and register the file type associations. Therefor, it requires an existing infrastructure with a file server, where the applications are streamed from, and a user management with group policies to take advantage of the login scripts.

To create virtual applications VMware ships a stand-alone executable called `Setup Capture` that makes a pre-scan snapshot before the program's installation and a post-scan snapshot afterwards. All changes that were made in this period are bundled to the virtual application. As the executable will run without contacting any server, it has to include everything that is related to the application. This includes not only the files and registry keys but also permissions and user groups that are required to run the program. Furthermore, defining dependencies to other packages that are loaded at runtime is also supported [66].

That said, there is no central management of applications, assignments and licenses like with App-V. Thus, the administrators have less control over the executables if they are handed out to users.

In spite of Microsoft's statements on the unsupported and unlicensed virtualization of Internet Explorer, ThinApp is able to run Internet Explorer 6 on Windows 7 by virtualizing it [2]. Furthermore, they offer a special component called `ThinDirect` which redirects predefined URLs to the appropriate web browser. So, they can switch between a virtual Internet Explorer 6 and a newer version that is installed natively without interrupting the user's work-flow [60]. This enables a company to use the older browser version for pages that are not compatible with the newer one.

5.2. Citrix Application Streaming

The Citrix Application Streaming is a part of the Citrix XenApp (formerly known as Presentation Server) product [53]. XenApp requires the Citrix Streaming Client software for executing and streaming the applications from a centralized server to the user's machine. Thus, XenApp requires an agent that is installed on the workstation and which handles the file system filtering in kernel mode. Certainly, the registry filtering is done in user mode which means that it also uses a hybrid approach like App-V [72].

It differs from App-V and ThinApp in that way, as it not only offers the streaming functionality but also can host applications remotely on a server. This feature is called `session virtualization` and comes into play when a client is not capable of running the virtual application locally because the device does not meet the requirements. Then, the XenApp Client executes the application entirely on the server and redirects keyboard and mouse inputs to it. That also enables different operating systems besides Windows to use the virtual applications remotely, as long as the XenApp Client is supported on the system. Thus, also Linux, Mac, thin clients, and even mobile devices are able to use the hosted programs as if they were running natively, including the direct access to printers and drivers of the local hardware [52].

The infrastructure includes

- the XenApp Profiler which creates the virtual applications and prepares them for streaming,
- a XenApp Farm that requires at least one server for the data store, a web interface,

and a management console,

- a license server that regulates the number of users that can concurrently use an application,
- the XenApp Plugins for Hosted and Streamed Apps (formerly known as Streaming Client) which provide a gateway to the published applications.

Furthermore, the applications can be accessed through a web interface which will execute the applications on the server and enable the before mentioned platform independence [53].

5.3. Symantec Workspace Virtualization

In 2007 Symantec aquired the company Altiris which developed a product called Software Virtualization Solution [43]. Afterwards, it was renamed to Symantec Workspace Virtualization (SWV) and is now integrated into several other products, such as the Symantec Endpoint Virtualization Suite, Symantec Workspace Streaming, or Software Management Solution.

SWV virtualizes applications by putting virtual software layers on top of the operating system. These layers can be activated and deactivated in order to make an application available to the user or not. To manage the layers it requires the Workspace Virtualization Agent to be installed on the workstations, which makes it a kernel mode product.

When an application's layer is activated, the agent publishes all files, registry keys, shortcuts, file type associations, and context menus like if the program was installed natively. So, SWV integrates an application more seamlessly into the system than App-V, ThinApp or XenApp. Thus, the user will find all files, registry keys and settings at their regular place as if the application would not have been virtualized. In order to accomplish this, SWV also relies on the redirection of files and registry keys to a protected place on the hard drive where the virtual application package is located. So, for example, the user can browse to a program directory in `C:\Program Files` in the Windows Explorer but SWV redirects the virtual directories to the protected virtual file system. This is not possible with neither of the other products because they make the virtual file system only visible to the specific virtual packages instead making it system-wide available.

SWV provides a writable user-specific layer that stores all the changes that a user makes to an application. Furthermore, a virtual package can contain a data layer that will capture all files that a user creates. This can be, for example, .doc files of Microsoft Word which will be redirected to the layer and which are hidden if the virtual application is deactivated. The special data layer has its advantage when the application layer has to be reset because this way the documents are not lost.

Symantec Workspace Virtualization can be used in a stand-alone mode or with one of the Symantec respectively Altiris server environments. Thus, not only streaming of applications is possible but also the integration of SWV into an existing network infrastructure because the agent provides different interfaces for managing the virtual software layers [44].

Unlike App-V, Symantec technically supports the virtualization of the Internet Explorer [3] but they also do not virtualize drivers and operating system components like service packs or Windows updates [44]. Certainly, the virtualization of the Internet Explorer may still be an unlicensed solution and thus is not supported by Microsoft [27].

6. Conclusion

The goal of this thesis was to research the application virtualization product of Microsoft and to find out whether it has the ability to fulfill the requirements of Munich's school network. There, Microsoft App-V is intended to solve most of the conflicts between applications and thus it was part of this thesis to analyze the options for its integration into the existing heterogeneous network. Furthermore, the virtualization of some varying software products that are already known to cause conflicts contributed to a categorization of software in order to summarize which applications are good candidates for virtualization and which cause problems.

The research showed that Microsoft App-V is a complex product that has to be understood when it comes to the virtualization of software. Thus, it is best to read the extensive documentation first before the sequencing is started. This thesis has given an overview of the App-V infrastructure and the main components as well as a summary on how the virtualization technically works. Therewith, a comprehensive view about drawbacks and opportunities of App-V and application virtualization in general is established.

The mere fact that applications can be executed without an installation is a huge benefit. Not only that the conflicts of files and registry keys are a thing of the past but also the option for streaming an application to the client on demand makes the administration of workstations easier. Thereby, updates can be deployed very fast by changing one file on a server and being sure that all users work with the newest version. This ensures a higher level of security and reduces the maintenance time that has to be dedicated when a software installation or removal causes problems.

For companies with a long list of applications the isolation of software that is achieved with virtualization is one promising way to solve conflicts. Certainly, it depends on the products whether all or at least most of the problems can be solved with application virtualization. So, this type of virtualization might not be appropriate for every network and it requires additional skills from administrators to solve the problems that occur in the virtualization process.

Furthermore, if the decision for one virtualization product has been made the migration from one product to another means to start at the very beginning because the virtual applications are not interchangeable. Thus, the long-term decision has to be deliberate and several products should be tested in researches and pilot experiments.

The integration of App-V into the school network has shown that it is possible to resign on the full infrastructure that is offered by the vendor and use the existing components without greater adaptations. Certainly, the pilot experiment has to show if everything works as planed and expected but it is already clear that all of the features that App-V offers are only available with the full infrastructure. It depends on the customer's need whether the absence is a loss or not.

For the end-user at home the before mentioned products are too expensive and some of them are only offered to enterprises. Many users did not experience application conflicts in the past because they only had a few programs installed on the computer. Today, there are much more software products available and users are working on different devices at differing locations. There, they would like to use the same software with their custom-made adjustments, too. More and more programs are offered as portable versions but most of the commercial software cannot be used this way. With application virtualization even the private users could generate portable programs that run from a USB flash drive and can be used on different computers without having them installed locally. Cameyo[1] is a freeware product that offers Windows users the fundamental features of application virtualization and might be a good start to provide a software even for the private sector.

The development and research on application virtualization is not at the end for a long time yet and thus the vendors will continue to improve their products. Microsoft's Server App-V shows that not only workstations are an interesting market but there is also a need on servers. Hence, there will be a better integration of virtual applications into the operating system in the future in order to allow interprocess communication especially on servers as well as on terminal services. The latter is an interesting area when it comes to thin clients and cloud computing. As the movement towards software as a service is upcoming, the abstract and isolated virtual applications are a good concept for this model. Furthermore, the workstations can be replaced by thin clients if applications can be executed completely on the server which reduces the maintenance-time enormously.

[1]http://www.cameyo.com

Appendix

A. App-V documentation for ZIB

Microsoft App-V für M@School

Windows 7 Client Entwicklungsprojekt

Zentrum für Informationstechnologie
im Bildungsbereich (ZIB)

Inhaltsverzeichnis

1 Einleitung **5**
1.1 Hintergrund . . . 5
1.2 Überblick Entwicklungsumgebung . . . 5
1.3 Hinweis . . . 6

2 Begriffsklärung **6**

3 Entwicklungsumgebung **8**
3.1 Host-System . . . 8
3.1.1 Hardware . . . 8
3.1.2 Software . . . 8
3.1.3 Benutzerkonten . . . 9
3.2 Virtuelle Maschinen . . . 9
3.2.1 Virtuelle Maschine für die Sequenzierung . . . 9
3.2.2 Virtuelle Maschine für den Test . . . 9

4 Konfiguration der Clients **10**
4.1 Verschiedene Modi der Verteilung . . . 10
4.1.1 Stand-Alone ohne Streaming . . . 10
4.1.2 Stand-Alone mit Streaming . . . 10
4.1.3 Publishing und Streaming . . . 11
4.2 Installation des App-V-Clients . . . 11
4.2.1 Stand-Alone ohne Streaming . . . 11
4.2.2 Stand-Alone mit Streaming . . . 12
4.2.3 Publishing und Streaming . . . 12
4.3 Ausblenden des virtuellen Laufwerks B . . . 13
4.4 Voraussetzungen für die Verteilung der MSI-Pakete . . . 14
4.5 Korrekte Wahl der Cache-Größe . . . 14
4.6 Integration von mobilen Geräten . . . 14
4.7 Management Console . . . 15
4.8 Funktionsweise von benutzerdefinierten Einstellungen . . . 15
4.9 Inhalte im Systemverzeichnis des App-V Clients . . . 16

5 Setup des Sequenzers **16**

6 Pakete erstellen und bearbeiten **17**
6.1 Vorüberlegungen . . . 17
6.2 Programm-Updates deaktivieren . . . 17
6.3 Neues Paket erstellen (Ablauf einer Sequenzierung) . . . 17
6.4 Update eines Pakets erstellen . . . 18
6.5 Verknüpfungen nachträglich ändern . . . 18

6.6 Abhängigkeiten zwischen Applikationen mit dem DSC eintragen . . 19

7 Installation eines virtuellen Pakets 19

7.1 Installation in der Entwicklungsumgebung 19

7.2 Installation mit dem VI-Client 20

8 Ablauf des Publishing-Vorgangs 20

8.1 Dynamische Applikationsliste 21

8.2 Nutzungsstatistiken (Reporting) 22

9 Deinstallation/Entzug einer Anwendung 22

1 Einleitung

1.1 Hintergrund

Die LHM möchte mit der Migration auf Windows 7 weitere Formen der Softwarebereitstellung testen, um vor allem Konflikte zwischen einzelnen Applikationen zu minimieren. Um zu untersuchen, welche Möglichkeiten eine Virtualisierungstechnik bieten kann und wie sich diese in das bestehende Konzept integrieren lässt, wurde im Rahmen einer Bachelorarbeit der Einsatz von Microsoft App-V näher untersucht und ein Vorschlag für die Integration implementiert.

Im Rahmen des LHM Windows 7 Client Entwicklungsprojekts wurde seitens T-Systems ein App-V Client V4.6 (64-bit) für den Windows 7 Client der LHM als VI-Paket paketiert. Dieser App-V Client ist für den stand-alone Modus konfiguriert, da die Serverumgebung den Einsatz der vollen Microsoft Infrastruktur momentan nicht ermöglicht. Allerdings wurden in der Bachelorarbeit Wege aufgezeigt, wie diese Infrastruktur nachgebildet und die Streaming-Funktionalität bereitgestellt werden kann. Außerdem wurde eine App-V Entwicklungsumgebung für die Erstellung und den Test virtualisierter Pakete aufgebaut, die hier näher beschrieben wird.

1.2 Überblick Entwicklungsumgebung

Für den Aufbau einer Entwicklungsumgebung wurde auf einem stand-alone Rechner ein Windows 7 64-bit installiert, welches als Host für virtuelle Maschinen der VMware Workstation dient. Diese sind für den Test und die Paketierung besonders gut geeignet, da mittels Snapshots die Maschine auf einen sauberen Zustand zurückgesetzt werden kann. Dies erspart die Neuinstallation des Rechners und ermöglicht auch die Sicherung von Zwischenständen während der Erstellung eines virtuellen Pakets.

Es wurden drei virtuelle Maschinen erzeugt:

1. App-V Sequencer: erstellt die virtuellen Pakete
2. App-V Test-Maschine: hier werden die virtuellen Pakete ausgeführt und getestet
3. Linux Server: dient mit einem Samba als Share für die Pakete, um das Streaming zu testen bzw. um die Pakete von der Sequencer-VM auf die Test-VM zu kopieren. Darüber hinaus ist auch ein Apache installiert, welcher dafür verwendet wurde, um die Publishing-Funktionalität nachzubilden.

Die beiden Erstgenannten sind ebenfalls Windows 7 64-bit Betriebssysteme, da dieses später in der Produktivumgebung auf den Clients zum Einsatz kommen wird. Damit es zu keinen Problemen bei der Verwendung der virtuellen Anwendungen kommt, sollten diese unter demselben Betriebssystem erstellt werden, unter dem sie auch später verwendet werden.

1.3 Hinweis

Momentan gibt es eine Unverträglichkeit zwischen App-V und dem bei der LHM eingesetzten Softwareverteilungssystem. Letztgenanntes steht mit dem von App-V verwendeten und gesperrten Laufwerksbuchstaben Q: in Konflikt und generiert einen Fehler.

Dieses Problem muss entweder seitens des Herstellers gelöst oder dadurch umgangen werden, dass ein anderer Laufwerksbuchstabe verwendet wird.

Letzteres ist die schnellste Lösung, daher wurde für den Windows-7-Piloten das Laufwerk B: gewählt, da der Softwareverteilungs-Client die Laufwerke A und B nicht beachtet.

Es ist davon auszugehen, dass keine Rechner mit mehr als einem Diskettenlaufwerk verwendet werden und auch sonst keine Belegung des Laufwerksbuchstabens B zu erwarten ist, weshalb die Lösung konfliktfrei sein sollte.

2 Begriffsklärung

App-V Client Anwendung auf dem Desktop eines Windows-basierten Rechners, der die Authentifizierung sowie Kommunikation mit einem Server (sofern im Online-Modus) übernimmt. Ohne diesen können die virtuellen Anwendungen nicht ausgeführt werden.

App-V Sequencer Anwendung zur Erstellung der virtuellen Applikationen. Die Sequenzierung kann sowohl über eine graphische Oberfläche als auch über Kommandozeile erfolgen.

DSC (Dynamic Suite Composition) Dieses Feature ermöglicht die Realisierung von Abhängigkeiten zwischen Applikationen und das Zusammenspiel von verschiedenen, normalerweise isolierten Anwendungen. Hierfür steht ein Tool namens „Microsoft Application Virtualization Dynamic Suite Composition Tool“ zur Verfügung, über das man mittels graphischer Oberfläche die Abhängigkeiten eines Pakets eintragen kann.

Feature Block 1 und 2 Der erste Feature Block wird auch „Primärer Feature Block“ genannt und stellt den Mindestinhalt eines virtuellen Anwendungspakets dar, um die Applikation auf dem Client ausführen zu können. Er wird beim Sequenzierungsvorgang gebildet und umfasst die am häufigsten verwendeten Anwendungskomponenten. Der zweite Feature Block (auch Sekundärer Feature Block) enthält den Rest der virtuellen Anwendung, der anschließend auf den Client übertragen wird.

ICO-Datei Symboldatei, die beim Publishing-Vorgang verwendet wird, um das Icon einer Verknüpfung zu bilden. Im Stand-Alone-Modus wird diese Datei bzw. der Ordner, in dem sich alle ICO-Dateien befinden, nicht benötigt.

OSD-Datei (Open Software Descriptor) XML-Datei, die Anweisungen enthält, wie die Applikation vom Streaming-Server abgerufen und in der virtuellen Umgebung ausgeführt werden soll.

Dort wird der Name, die Version und die eindeutige GUID der Applikation festgelegt. Weiterhin befindet sich dort die Angabe zum Speicherort der SFT-Datei und die Betriebssystem-Kompatibilität (Windows XP/7 32-bit/64-bit).

Package-GUID Eindeutiger Identifikationsstring (z. B. „3D726A15-719F-4952-BB74-AE7ED06D35F5“) unter dem die Applikation registriert wird. Mit dieser ID werden auch die benutzerspezifischen Einstellungen im Dateisystem abgelegt.

Publishing-Vorgang Kommunikationsvorgang zwischen Client und Server, um eine XML-Datei vom Server abzurufen, welche die Anwendungen enthält, die der Anwender nutzen darf.

Report-Datei Nicht zu verwechseln mit dem Reporting-Feature beim Publishing. Diese XML-Datei enthält das Log des Sequenziervorgangs und kann bei Problemen konsultiert werden, um herauszufinden, welche Verzeichnisse wegen der Ausnahmeliste nicht hinzugefügt werden konnten oder wenn Einstellungen erkannt wurden, die App-V nicht unterstützt (z. B. Shellerweiterungen).

Reporting-Feature Dieses Feature ist Teil des Publishing-Vorgangs und überträgt Nutzungsstatistiken (Name der Anwendung, Username und Domäne, Zeitpunkt des Anwendungsstarts, Zeitpunkt der Beendung) per POST-Request an den Server.

SFT-Datei Enthält die sequenzierte(n) Anwendung(en), welche in Streamingblöcke eingeteilt ist.

SFTMIME Eine Befehlszeilenschnittstelle für das Verwalten von Anwendungen, Verknüpfungen, Dateierweiterungen und Veröffentlichungsservern. Hiermit können die Schritte des MSI-Pakets eigenständig geskriptet werden.

SPRJ-Datei XML-Projektdatei des Sequencers, welche alle Projektkonfigurationen (MSI erstellen, Paket komprimieren, ...) und Ausnahmeverzeichnisse enthält. Hier kann im Nachhinein eine Ausnahme entfernt werden, um eine Datei nachträglich zum Paket hinzuzufügen.

Streaming Blockweise Übertragung der virtuellen Applikationsdatei (SFT) vom Server zum Client. Zuerst wird der Primäre Feature Block übertragen, um die Anwendung so schnell wie möglich betriebsbereit zu bekommen. Dies ermöglicht die zentrale Verwaltung von SFTs und erspart das Kopieren der virtuellen Applikation auf den Client. Der Streaming-Vorgang sorgt dafür, dass der lokale Cache befüllt wird.

3 Entwicklungsumgebung

3.1 Host-System

3.1.1 Hardware

Modell: HP Compaq DC7900 CMT
RAM: 4 GB
HDD: 150 GB

Damit die 64-bit Version von Windows virtualisiert werden kann, musste im BIOS die Virtualisierungsfunktion (VT-x) aktiviert werden.

3.1.2 Software

Betriebssystem
Microsoft Windows 7 Enterprise 64-bit, SP1, Deutsch

Konfiguration
Standardinstallation, eine System-Partition, Arbeitsgruppe: `WORKGROUP`

Sonstige Software

- Adobe Reader X
- 7-Zip 9.20

- Mozilla Firefox
- VMware Workstation Version 7

3.1.3 Benutzerkonten

- Administrator-Konto ist deaktiviert (Windows Voreinstellung)
- nimda (mit dem Standard-Passwort der LHM) als Administrator; gilt auch für virtuelle Maschinen
- manuel (Manuel Söhner)

3.2 Virtuelle Maschinen

Es wird die Software VMware Workstation mit einer LHM-Lizenz eingesetzt, da VirtualPC für die Verwendung eines 64-bit Betriebssystems nicht ausgelegt ist.

Der Datenaustausch zwischen dem Host und den VMs erfolgt über VMware Shared Folders beziehungsweise über den Share des Linux-Servers, der ebenfalls in einer virtuellen Maschine installiert wurde.

3.2.1 Virtuelle Maschine für die Sequenzierung

Name: Windows7SEQ
Beschreibung: Windows 7 Enterprise 64-bit, SP1, Deutsch
Standardinstallation mit App-V V4.6 SP1, Sequenzer
Snapshot vorhanden, Maschine kann nach jedem Sequenzierungsvorgang wieder auf den Ausgangszustand zurückgesetzt werden.

Diese VM benutzt eine zweite Partition (B:), welche mit NTFS formatiert wurde. Dieses Laufwerk wird von App-V verwendet, um die Dateien zu speichern, welche eine Anwendung installiert hat. Hier hat man vollen Zugriff auf das Laufwerk.

3.2.2 Virtuelle Maschine für den Test

Name: Windows7_Test
Beschreibung: Windows 7 Enterprise 64-bit, SP1, Deutsch
Standardinstallation mit App-V V4.6 SP1, Client
Snapshot vohanden, Maschine kann nach jedem Test wieder auf den Ausgangszustand zurückgesetzt werden.

In dieser VM muss keine extra Partition angelegt werden. Das virtuelle Laufwerk (B:) wird von App-V erzeugt und ist vor Zugriffen geschützt.

Zwischenzeitlich wurde der App-V Client auch im Online-Modus getestet, um die Infrastruktur von Microsoft nachzubilden, allerdings wird diese von der LHM für den Piloten zunächst nicht angestrebt. Daher ist der Client im stand-alone Modus konfiguriert und setzt somit voraus, dass die virtuellen Pakete über den Shared Folder von VMware oder den Linux Samba-Share auf die Test-Maschine kopiert und über das MSI installiert werden.

4 Konfiguration der Clients

4.1 Verschiedene Modi der Verteilung

4.1.1 Stand-Alone ohne Streaming

Da der Client im Offline-Modus arbeitet, müssen die Applikationspakete (MSI und SFT) manuell oder über eine Softwareverteilung verteilt werden. Hierfür erzeugt der Sequencer ein MSI, welches bei der Installation

1. Dateierweiterungen registriert,
2. Verknüpfungen erstellt,
3. das Send-to-Menü erweitert und
4. den lokalen Cache mit der SFT befüllt.

4.1.2 Stand-Alone mit Streaming

Der Client arbeitet im Online-Modus und streamt beim Start der Anwendung das SFT von einem Server auf den Client, sofern sich diese noch nicht im Cache befindet. Dabei wird zunächst der Feature Block 1 übertragen, welcher die für den Start der Applikation relevanten Teile enthält, um einen schnellen Start gewährleisten zu können. Anschließend wird der Feature Block 2 sukzessive nachgeladen.

Als Übertragungsprotokoll kann HTTP(S), SMB oder RTSP verwendet werden. Je nachdem muss die zu streamende SFT-Datei über einen geeigneten Dienst zur Verfügung gestellt werden.

4.1.3 Publishing und Streaming

Bei dieser Methode wird komplett auf das MSI verzichtet und eine zentrale XML-Datei verwendet, welche die verfügbaren Applikationen enthält.

4.2 Installation des App-V-Clients

Für alle Modi müssen folgende Systemvoraussetzungen[1] geschaffen sein:

- Microsoft Visual C++ 2005 SP1 Redistributable Package (x86)
- Microsoft Core XML Services (MSXML) 6.0 SP1 (x86)
- Microsoft Application Error Reporting
- Microsoft Visual C++ 2008 SP1 Redistributable Package (x86)

Diese werden bei der Verwendung der setup.exe automatisch installiert, bei der Verwendung der MSI müssen diese separat installiert werden.

4.2.1 Stand-Alone ohne Streaming

```
setup.exe /s /v"/qn REQUIREAUTHORIZATIONIFCACHED=\"False\"
ALLOWINDEPENDENTFILESTREAMING=\"True\" MINFREESPACEMB=\"5000\"
SWIFSDRIVE=\"B\""
```

Auflistung 1: Installationsparameter

```
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
Client\Network" /v LimitDisconnectedOperation /t REG_DWORD
/d 0

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
Client\Network" /v Online /t REG_DWORD /d 0

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
Client\Permissions" /v ToggleOfflineMode /t REG_DWORD /d 0
```

Auflistung 2: Registry-Änderungen

[1] http://technet.microsoft.com/en-us/library/cc843822.aspx

4.2.2 Stand-Alone mit Streaming

```
1. setup.exe /s /v"/qn REQUIREAUTHORIZATIONIFCACHED=\"True\"
ALLOWINDEPENDENTFILESTREAMING=\"True\" MINFREESPACEMB=\"5000\"
SWIFSDRIVE=\"B\""
```

Auflistung 3: Installationsparameter

```
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
Client\Configuration" /v ApplicationSourceRoot /t REG_SZ
/d "\\SERVERNAME\SHARE"

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
Client\Network" /v LimitDisconnectedOperation /t REG_DWORD
/d 1

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
Client\Network" /v Online /t REG_DWORD /d 1

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
Client\Permissions" /v ToggleOfflineMode /t REG_DWORD /d 0
```

Auflistung 4: Registry-Änderungen

4.2.3 Publishing und Streaming

```
1. setup.exe /s /v"/qn REQUIREAUTHORIZATIONIFCACHED=\"True\"
ALLOWINDEPENDENTFILESTREAMING=\"True\" MINFREESPACEMB=\"5000\"
SWIFSDRIVE=\"B\""
```

Auflistung 5: Installationsparameter

```
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
Client\Configuration" /v ApplicationSourceRoot /t REG_SZ
/d "\\SERVERNAME\SHARE"

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
Client\Network" /v LimitDisconnectedOperation /t REG_DWORD
/d 1

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
Client\Network" /v Online /t REG_DWORD /d 1
```

```
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
Client\Permissions" /v ToggleOfflineMode /t REG_DWORD /d 0

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
DC Servers\Schulserver" /v ID /t REG_DWORD /d 1

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
DC Servers\Schulserver" /v Type /t REG_SZ /d "HTTP"

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
DC Servers\Schulserver" /v Host /t REG_SZ /d "SERVERNAME"

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
DC Servers\Schulserver" /v Path /t REG_SZ /d "/applist.php"

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
DC Servers\Schulserver" /v Port /t REG_DWORD /d 80

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
DC Servers\Schulserver" /v Reporting /t REG_DWORD /d 0

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
DC Servers\Schulserver" /v Refresh /t REG_DWORD /d 1

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SoftGrid\4.5\
DC Servers\Schulserver" /v Period /t REG_DWORD /d 0
```

Auflistung 6: Registry-Änderungen

4.3 Ausblenden des virtuellen Laufwerks B

Um das zugriffsgeschützte Laufwerk nicht mehr im Explorer anzeigen zu lassen, muss ein Registry-Key gesetzt bzw. die Group Policy für den User angepasst werden.

```
reg add "HKEY_CURRENT_USER\Software\Microsoft\Windows\
CurrentVersion\Policies\Explorer" /v NoDrives /t REG_DWORD
/d 2
```

Auflistung 7: Registry-Key zum Verstecken des Laufwerks B

4.4 Voraussetzungen für die Verteilung der MSI-Pakete

Bei der Verteilung sollte darauf geachtet werden, dass sämtliche Voraussetzungen erfüllt sind, bevor die Anwendungen installiert werden.

Dies setzt voraus, dass die Pakete in einer bestimmten Reihenfolge installiert werden. Zunächst müssen die Voraussetzungen für den App-V-Client installiert werden (siehe oben: Systemvoraussetzungen).

Außerdem sollte sichergestellt werden, dass der App-V-Client vor den virtuellen Applikationen installiert wird, da diese sonst nicht verarbeitet werden können.

Hierfür bietet der AE-Manager eine Sortierreihenfolge an, über die festgelegt werden kann, welche Reihenfolge durch den VI-Client bei der Installation einzuhalten ist.

4.5 Korrekte Wahl der Cache-Größe

Der Cache sollte so konfiguriert werden, dass er wachsen kann, bis die Festplatte fast voll ist. Dies senkt das Risiko, dass der Cache voll läuft und die am längsten nicht mehr verwendeten Applikationen verdrängt werden.

Bei „Stand-Alone ohne Streaming“ führt dies zu einem Problem, da der Cache im Nachhinein mit der Anwendung nicht mehr initialisiert werden kann. Dies führt dazu, dass die Anwendung nicht mehr verwendbar ist.

4.6 Integration von mobilen Geräten

Die Variante von Stand-Alone ohne Streaming benötigt bei Notebooks keinerlei zusätzliche Anpassungen, da sich die virtuelle Applikation komplett im Cache befindet, nachdem sie installiert wurde.

Beim Einsatz von Streaming muss man beachten, dass das Notebook nur dann offline betrieben und die Applikationen genutzt werden können, wenn der Cache bereits gefüllt ist.

Dies geschieht bei der Standardinstallation des App-V-Clients erst bei der ersten Verwendung einer Anwendung. Sollen stattdessen sämtliche Applikationen sofort bei der ersten Anmeldung am Notebook in den Cache geladen werden, müssen in der Registry unter

```
HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\SoftGrid\4.
5\Client\Configuration
```

die Keys `AutoLoadTriggers` und `AutoLoadTarget` gemäß der Dokumentation[2] angepasst werden. Bei Notebooks würde man AutoLoadTarget auf den Wert „2“ setzen und AutoLoadTriggers vermutlich bei „5“ belassen.

4.7 Management Console

```
C:\Program Files (x86)\Microsoft Application Virtualization Client\
SftCMC.msc
```

Hier können, sofern die Berechtigungen vorliegen, die Applikationen, Dateierweiterungen und Publishing-Server verwaltet werden.

Eine Applikation kann

1. auf Aktivität/Ausführung hin überprüft,
2. über den Ladezustand im Cache befragt,
3. gelöscht,
4. repariert,
5. zurückgesetzt,
6. vor dem Verdrängen aus dem Cache geschützt,
7. in den Cache importiert und
8. aus dem Cache entladen

werden. Außerdem lässt sich hier nachträglich eine Verknüpfung oder Dateierweiterung einfügen sowie bestehende bearbeiten.

Der Veröffentlichungsserver wird in der Regel bei der Installation eingetragen und nachträglich nicht mehr verändert werden.

In den Eigenschaften können Berechtigungen, Speicherorte, Anzeigeverhalten und die Protokollierungsstufe festgelegt werden. Allerdings werden auch diese Einstellungen einmal definiert und bei der Installation per Registry-Werte festgeschrieben.

[2]`http://technet.microsoft.com/en-us/library/dd464849.aspx`

4.8 Funktionsweise von benutzerdefinierten Einstellungen

An der virtuellen Applikation selbst werden keine Änderungen vorgenommen, wenn der Anwender Einstellungen vornimmt. Stattdessen wird im Profil im Ordner

```
%AppData%\SoftGrid Client
```

für jede Anwendung eine UsrVol_sftfs_v1.pkg-Datei erstellt, welche die Anpassungen enthält.

Bei der Ausführung des Programms werden somit immer die benutzerspezifischen Einstellungen hinzugeladen und überschreiben die Systemvorgaben in der virtuellen Applikation.

Somit ist sichergestellt, dass die virtuelle Applikation intakt bleibt und jeder Anwender individuell mit der Applikation arbeiten kann.

Sollte es im Betrieb zu Problemen mit der Anwendung kommen, kann das Profil des Anwenders repariert werden, wodurch sämtliche Einstellungen verloren gehen. Alternativ lässt sich (über die Management Console oder durch gezieltes Löschen des Verzeichnisses aus dem Dateisystem) für eine bestimmte Anwendung die Datei zurücksetzen. Beim manuellen Vorgehen muss jedoch zunächst die Package-GUID der Applikation ausfindig gemacht werden.

Außerdem werden im Anwendungsverzeichnis auch die Icons zwischengespeichert, die von den Verknüpfungen benötigt werden.

4.9 Inhalte im Systemverzeichnis des App-V Clients

Im Verzeichnis

```
C:\Program Data\Microsoft\Application Virtualization Client
```

befindet sich die Log-Datei (sftlog.txt), welche bei Problemen darüber Aufschluss geben kann, was die Ursache dafür sein könnte.

Hier werden außerdem die OSD-Dateien und Icons gecachet.

5 Setup des Sequenzers

Wie beim App-V-Client müssen die Systemvoraussetzungen installiert werden. Diese werden nur bei der Verwendung der Setup-Datei automatisch installiert.

Beim MSI müssen diese separat installiert werden. Außer der Einstellung des Laufwerks „B“ muss bei der Installation auf nichts weiter geachtet werden.

6 Pakete erstellen und bearbeiten

6.1 Vorüberlegungen

Sollen mehrere Versionen (lokal und/oder virtualisiert) parallel betrieben werden, muss darauf geachtet werden, dass die Verknüpfungen eindeutig sind.

Am besten erzeugt man im Startmenü für alle virtuellen Applikationen einen Eintrag im Ordner „Virtualisierung“, damit diese nicht mit lokal installierten Programmen kollidieren.

Für Verknüpfungen auf dem Desktop muss ein ähnliches Namensschema gefunden werden.

Bei den Dateierweiterung ist zu bedenken, dass diese immer nur einem Programm zugeordnet sein können.

Wird ein Paket per MSI über die Softwareverteilung installiert, überschreibt die zuletzt installierte Anwendung die Verknüpfungen und Dateierweiterung.

6.2 Programm-Updates deaktivieren

Da ein aktives Update immer in den App-V Benutzereinstellungen gespeichert wird und nie die virtuelle Anwendung an sich betrifft, würde die Datei durch das Update unnötig vergrößert werden.

6.3 Neues Paket erstellen (Ablauf einer Sequenzierung)

Die „Best practices to use for sequencing in Microsoft App-V (SoftGrid)“[3] sind ein guter Einstieg zur Vermeidung von Fehlern beim Sequenzieren.

1. Installationsquellen nach `C:\packages` kopieren (oder einen anderen vom Monitoring ausgenommenen Ordner)

2. Standardanwendung

[3]`http://support.microsoft.com/kb/932137/en-us`

3. Installer auswählen oder manuelle Installation verwenden
4. Namen vergeben (`Hersteller_Produktname_Version_[MNT|VFS]`) MNT, falls die Installation nach B: erfolgt, sonst VFS (Virtual File System)
5. Programm, Updates, etwaige Addons installieren
6. Monitoring beenden
7. Programm konfigurieren (Einstellungen, Updates deaktivieren!)
8. Verknüpfungen kontrollieren, gegebenenfalls unnötige löschen
9. Dateierweiterungen kontrollieren
10. Feature Block 1 beim Streaming erstellen
11. Beenden und speichern lassen
12. Paket bearbeiten und Registry überprüfen, unnötige Einträge entfernen
13. Namen gegebenenfalls anpassen (z. B. Bei Mozilla-Produkten), um keine Namenskonflikte bei der Installation des MSI zu erhalten

6.4 Update eines Pakets erstellen

Hierfür bestehen zwei Möglichkeiten:

1. Erstellen eines neuen Pakets, welches das Update enthält
2. Erweitern eines bestehenden Pakets

Die erste Variante verläuft wie ein normaler Sequenziervorgang einer neuen Applikation, bei der zweiten öffnet man das Paket und wählt „Anwendung im vorhandenen Paket aktualisieren“.

Hierbei wird die Anwendung wieder nach `B:\` entpackt und kann anschließend durch ein Setup oder Programm-interne Update-Routinen aktualisiert werden.

6.5 Verknüpfungen nachträglich ändern

- Paket öffnen und „Hinzufügen einer neuen Anwendung“ wählen
- Der Prozess ist derselbe wie beim Sequenzieren, allerdings klickt man immer nur auf Weiter und fügt keine neue Anwendung hinzu. Am Ende bearbeitet man die Verknüpfungen wie gehabt.

- Dabei bleiben die vorher definierten Abhängigkeiten zu anderen Paketen erhalten
- Die Abhängigkeiten (z. B. Java) müssen vorher nicht lokal installiert werden, da die Anwendung nicht gestartet werden muss. Anderenfalls (bei einem Update) müssten sie das, da die Applikation sonst nicht korrekt funktionieren wird.

6.6 Abhängigkeiten zwischen Applikationen mit dem DSC eintragen

Das Bearbeiten der OSD-Datei via Texteditor ist möglich, allerdings funktioniert dieses Vorgehen nur bei „Publishing und Streaming“, bei dem kein MSI zum Einsatz kommt.

Anderenfalls muss der Sequencer oder das DSC verwendet werden. Da der Weg über DSC am einfachsten ist und hierbei keine Fehler (falsche Pfadangaben) unterlaufen können, ist dieses Vorgehen zu bevorzugen.

1. Microsoft Application Virtualization Dynamic Suite Composition Tool[4] starten
2. Als `Package root(s)` den Ort angeben, unter dem die Applikationspakete gespeichert sind
3. `Primary Package` über das DropDown-Menü wählen
4. Auf der linken Seite aus der Liste eines oder mehrere `Secondary Packages` mittels `Add` hinzufügen
5. Falls bei den Dependencies die Checkbox angehakt ist, ist die Abhängigkeit für den Betrieb der Software unverzichtbar („mandatory“)
6. Beim Speichern über `Save` wird die OSD-Datei der Primary-Applikation und dessen MSI-Paket angepasst.

[4] `http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=6655`

7 Installation eines virtuellen Pakets

7.1 Installation in der Entwicklungsumgebung

1. Die virtuelle Maschine „Windows7_Test“ auf einen sauberen Snapshot zurücksetzen
2. App-V Paket über den VMware Shared Folder oder vom Linux Samba-Share holen und lokal ablegen. Die Installation direkt vom Freigabeverzeichnis ist nicht möglich, da App-V die virtuelle Applikationsdatei (SFT-Datei) nicht laden kann.
3. MSI-Datei ausführen, um die Anwendung zu registrieren und Verknüpfungen sowie Dateierweiterungen zu erstellen
4. Anwendung mit verschiedenen Benutzerkonten (Administrator und eingeschränktes Konto) testen. Darunter fällt das Vornehmen von Konfigurationseinstellungen an der Anwendung, welche über den Neustart der Anwendung hinaus erhalten bleiben sollen. Ist dies nicht der Fall, muss das Paket im Sequencer bearbeitet und der Haken bei „Sicherheitsbeschreibungen durchsetzen“ entfernt werden.

7.2 Installation mit dem VI-Client

1. Die virtuelle Anwendung wird wie ein normales MSI-Paket im AE-Manager angelegt.
2. Es muss allerdings im Konfigurationseditor die Option MANAGEDESKTOPICONS deaktiviert werden, da anderenfalls bei einer Deinstallation des Pakets die Verknüpfungen nicht gelöscht werden.
3. Zusätzlich sollte die Sortierreihenfolge der virtuellen Applikation angepasst werden, damit diese in der Abarbeitung erst am Ende installiert werden.

8 Ablauf des Publishing-Vorgangs

1. Client verbindet sich per HTTP(S) zum konfigurierten Publishing-Server und fragt die Applikationsveröffentlichungsdatei (XML) per GET-Request an
2. Falls der Zugriff auf den Server eine NTLM-Authentifizierung (selbes Apache Modul wie bei WebDAV) fordert, wird in einem Challenge-Response-

Verfahren die Authentizität des Users geklärt. Dies ermöglicht auch eine dynamische Generierung der XML-Applikationsliste, da nur hier der Username und die Domäne des Anwenders übertragen werden, anhand derer man entscheiden kann, welche Anwendungen er nutzen darf.

3. Client schickt (falls aktiviert) die Reporting-XML per POST-Request zum Server, der diese auswerten kann

4. Der Client erzeugt auf Basis der XML-Datei die eingetragenen Verknüpfungen und Dateierweiterungen

8.1 Dynamische Applikationsliste

Die angesprochene XML-Datei lässt sich dynamisch mit einem Skript in Perl/PHP generieren. Dazu könnten die virtuellen Anwendungen des Users aus dem LDAP ausgelesen und ein entsprechendes XML-Template für die Applikation von diesem Skript eingebunden werden. Durch die Authentifikation über NTLM bekommt man in der Server-Variable `REMOTE_USER` den Benutzernamen, anhand dessen der spezifische LDAP-Query erzeugt werden kann.

Das Template für die Applikation kann auf Basis der Manifest.xml-Datei erstellt werden, die vom Sequencer erzeugt wird und in jedem Applikationsverzeichnis vorhanden ist. Es muss lediglich mit einem Parser alles zwischen dem `<APPLIST>`-Element extrahiert werden, das dann als Template dient.

```
<APP NAME="FileZilla Client" VERSION="3.5.2.0"
  ICON="%SFT_MIME_SOURCE%/FileZilla Icons/FileZilla.ico"
  OSD="%SFT_MIME_SOURCE%/FileZilla.osd">
  <SHORTCUTLIST>
    <SHORTCUT LOCATION="%CSIDL_DESKTOPDIRECTORY%"
      ICON="%SFT_MIME_SOURCE%/FileZilla Icons/FileZilla.ico"
      PARAMETERS="" DISPLAY="FileZilla Client"/>
    <SHORTCUT LOCATION="%CSIDL_PROGRAMS%\FileZilla FTP Client"
      ICON="%SFT_MIME_SOURCE%/FileZilla Icons/FileZilla.ico"
      PARAMETERS="" DISPLAY="FileZilla Client"/>
  </SHORTCUTLIST>
  <TYPELIST/>
</APP>
```

Auflistung 8: Manifest von FileZilla als Template

Diese Templates müssen schließlich in ein neues Format gebracht werden:

```
<DESKTOPCONFIG>
    <POLICY MANAGEDDESKTOP="TRUE" REPORTING="FALSE">
      <REFRESH ONLOGIN="TRUE" PERIOD="60"/>
    </POLICY>
  <APPLIST>
    [ ... TEMPLATES ... ]
  </APPLIST>
</DESKTOPCONFIG>
```

Auflistung 9: Manifest von FileZilla als Template

Anschließend wird dieser gesamte Inhalt an den anfragenden App-V Client zurückgeliefert.

8.2 Nutzungsstatistiken (Reporting)

Wird das XML-Gerüst der obigen Auflistung dahingehend angepasst, dass

`REPORTING="TRUE"`

gesetzt ist, können vom Client Statistiken abgefragt werden. Alternativ kann das Reporting auch auf dem Client in der Registry konfiguriert werden, ist über die zentrale XML-Datei allerdings deutlich einfacher zu steuern.

Der Client schickt dann mit derselben Anfrage beim Publishing-Vorgang nach dem GET (Anfrage der XML-Datei) ein POST, in dessen Nutzlast sich ein XML-String befindet, welcher die Nutzungsstatistiken abbildet. Dieses XML muss entsprechend geparst und die Daten gemäß der Datenschutzrichtlinien gespeichert werden.

9 Deinstallation/Entzug einer Anwendung

Bei der Deinstallation werden die Verknüpfungen und Dateierweiterungen entfernt, allerdings bleibt die Applikation im Cache.

Dies ermöglicht beim „Publishing und Streaming“ jedoch das unerlaubte Ausführen des Programms auch nach dem Entzug, wenn der Applikationsname für den Aufruf bekannt ist. Dadurch könnte ein findiger Anwender eine neue Verknüpfung erzeugen und den Aufruf der Applikation damit nachbilden.

Der Zugriff auf die Server-seitig gespeicherte SFT-Datei kann jedoch mittels ACL unterbunden werden. Dazu muss für jede Anwendung eine Gruppe existieren, der ein User zugewiesen wird, falls er das Recht auf Verwendung eingeräumt bekommen soll.

Im Dateisystem weist man dem Ordner des virtuellen Pakets die Gruppe zu und setzt die Zugriffsrechte für die Gruppe auf read-only. Die Rechte für „other“ sollten komplett verweigert werden (`chmod g=rx,o= dirname`).

Bibliography

[1] Augusto Alvarez. *Getting Started with Microsoft Application Virtualization 4.6*. Packt Publishing, January 2011.

[2] Peter Bjork. Step by step instructions on how to run internet explorer 6 on windows 7. `http://blogs.vmware.com/thinapp/2010/01/step-by-step-instructions-on-how-to-thinapp-internet-explorer-6-to-work-on-windows-7.html`, January 2010. [Online; accessed 24-October-2011].

[3] Karl Bunnell. Running ie6, ie7 and ie8 side-by-side using symantec workspace virtualization. `http://www.symantec.com/connect/articles/running-ie6-ie7-and-ie8-side-side-using-symantec-workspace-virtualization`. [Online; accessed 5-November-2011].

[4] J. P. Buzen and U. O. Gagliardi. The evolution of virtual machine architecture. In *Proceedings of the June 4-8, 1973, national computer conference and exposition*, AFIPS '73, pages 291–299, New York, NY, USA, 1973. ACM.

[5] Focus Consulting. Focus solution profile: Microsoft application virtualization. `http://download.microsoft.com/download/B/1/3/B13CA428-05C9-4327-8BB1-0D69302A5B24/Focus_SP_Microsoft_Application_Virtualization.pdf`, 2008. [Online; accessed 26-October-2011].

[6] Randy Cook. Application virtualization: Under the hood. `http://www.brianmadden.com/blogs/videos/archive/2009/08/25/A-Geeks-Look-at-How-Application-Virtualizaton-Works-Under-the-Hood_2C00_-a-video-from-BriForum-2009.aspx`, 2009. [Online; accessed 26-October-2011].

[7] Microsoft Corporation. 64-bit versions of windows do not support 16-bit components, 16-bit processes, or 16-bit applications. `http://support.microsoft.com/kb/896458/en-us`. [Online; accessed 04-November-2011].

[8] Microsoft Corporation. Application virtualization client installer command-line parameters. `http://technet.microsoft.com/en-us/library/cc843737.aspx`. [Online; accessed 10-October-2011].

[9] Microsoft Corporation. Application virtualization server-based scenario overview. `http://technet.microsoft.com/en-us/library/cc843686.aspx`. [Online; accessed 29-October-2011].

[10] Microsoft Corporation. Application virtualization system requirements. `http://technet.microsoft.com/en-us/library/cc843853.aspx`. [Online; ac-

cessed 29-October-2011].

[11] Microsoft Corporation. Best practices to use for sequencing in microsoft app-v (softgrid). `http://support.microsoft.com/kb/932137/en-us/`. [Online; accessed 15-October-2011].

[12] Microsoft Corporation. Changing environment variables. `http://msdn.microsoft.com/en-us/library/windows/desktop/ms682009%28v=vs.85%29.aspx`. [Online; accessed 03-November-2011].

[13] Microsoft Corporation. Csidl. `http://msdn.microsoft.com/en-us/library/windows/desktop/bb762494%28v=vs.85%29.aspx`. [Online; accessed 14-October-2011].

[14] Microsoft Corporation. File management functions. `http://msdn.microsoft.com/en-us/library/windows/desktop/aa364232%28v=VS.85%29.aspx`. [Online; accessed 03-November-2011].

[15] Microsoft Corporation. File system filter driver classes and class guids. `http://msdn.microsoft.com/en-us/library/windows/hardware/ff540394%28v=vs.85%29.aspx`. [Online; accessed 03-November-2011].

[16] Microsoft Corporation. File system filter drivers. `http://msdn.microsoft.com/en-us/windows/hardware/gg462968`. [Online; accessed 03-November-2011].

[17] Microsoft Corporation. Filter manager concepts. `http://msdn.microsoft.com/en-us/library/windows/hardware/ff541610%28v=vs.85%29.aspx`. [Online; accessed 03-November-2011].

[18] Microsoft Corporation. Filtering registry calls. `http://msdn.microsoft.com/en-us/library/windows/hardware/ff545879%28v=vs.85%29.aspx`. [Online; accessed 03-November-2011].

[19] Microsoft Corporation. How to sequence an application. `http://technet.microsoft.com/en-us/library/cc817128.aspx`. [Online; accessed 15-October-2011].

[20] Microsoft Corporation. How to use dynamic suite composition. `http://technet.microsoft.com/en-us/library/cc843662.aspx`. [Online; accessed 22-October-2011].

[21] Microsoft Corporation. Microsoft application virtualization (app-v). `http://www.microsoft.com/en-us/windows/enterprise/products-and-technologies/virtualization/app-v.aspx`. [Online; accessed 27-October-2011].

[22] Microsoft Corporation. New uac technologies for windows vista. `http://msdn.microsoft.com/en-us/library/bb756960.aspx`. [Online; accessed 26-October-2011].

[23] Microsoft Corporation. Overview of application virtualization. `http://technet.microsoft.com/en-us/library/ee958112.aspx`. [Online; accessed 29-October-2011].

[24] Microsoft Corporation. Planning for server security. `http://technet.microsoft.com/en-us/library/dd351448.aspx`. [Online; accessed 29-October-2011].

[25] Microsoft Corporation. Process monitor. `http://technet.microsoft.com/en-us/sysinternals/bb896645`. [Online; accessed 09-November-2011].

[26] Microsoft Corporation. Registry functions. `http://msdn.microsoft.com/en-us/library/ms724875%28v=VS.85%29.aspx`. [Online; accessed 12-October-2011].

[27] Microsoft Corporation. Running multiple versions of internet explorer on a single instance of windows is unsupported. `http://support.microsoft.com/kb/2020599/en-us`. [Online; accessed 5-November-2011].

[28] Microsoft Corporation. Support for .net in microsoft application virtualization 4.5 (app-v). `http://technet.microsoft.com/en-us/appvirtualization/dd146065`. [Online; accessed 27-October-2011].

[29] Microsoft Corporation. Windows 95 architecture components. `http://technet.microsoft.com/en-us/library/cc751120.aspx`. [Online; accessed 01-November-2011].

[30] Microsoft Corporation. Side-by-side execution of the .net framework. `http://msdn.microsoft.com/en-us/library/ms994410.aspx`, 2002. [Online; accessed 28-October-2011].

[31] Microsoft Corporation. General overview of win32s. `http://support.microsoft.com/kb/83520/en-us`, April 2004. [Online; accessed 16-October-2011].

[32] Microsoft Corporation. Microsoft completes acquisition of softricity. `http://www.microsoft.com/presspass/press/2006/jul06/07-17softricitypr.mspx`, June 2006. [Online; accessed 3-October-2011].

[33] Microsoft Corporation. Working with the appinit_dlls registry value. `http://support.microsoft.com/kb/197571/en-us`, 2006. [Online; accessed 12-October-2011].

[34] Microsoft Corporation. App-v application publishing and client interaction. `http://download.microsoft.com/download/f/7/8/f784a197-73be-48ff-83da-4102c05a6d44/APP-V/AppPubandClientInteraction.docx`, 2008. [Online; accessed 31-October-2011].

[35] Microsoft Corporation. Application virtualization cost reduction study. `http://download.microsoft.com/documents/France/windows/2010/`

windows7/App-V%20Cost%20Reduction%20Study.pdf, 2009. [Online; accessed 25-October-2011].

[36] Microsoft Corporation. Microsoft application virtualization file format specification. http://download.microsoft.com/download/E/B/9/EB967B04-2F6E-4DB2-B6A9-72782D3392E1/App-V_file_format_v1.doc, February 2009. [Online; accessed 04-October-2011].

[37] Microsoft Corporation. Application virtualization 4.6 for windows server 2008 r2 remote desktop services. http://download.microsoft.com/download/2/5/E/25EEFF4E-A81A-464F-9AB1-98FA1EF755AA/App-V%20Remote%20Desktop%20Services.docx, 2010. [Online; accessed 30-October-2011].

[38] Microsoft Corporation. Microsoft application virtualization volume format specification. http://download.microsoft.com/download/7/7/D/77DC8335-89FF-4054-96FE-52D1667EECC0/Application%20Virtualization%20Volume%20Format%20Specification.exe, May 2010. [Online; accessed 15-October-2011].

[39] Microsoft Corporation. Solutions for virtualizing internet explorer. http://www.microsoft.com/download/en/details.aspx?id=9242, 2010. [Online; accessed 28-October-2011].

[40] Microsoft Corporation. Using the runasinvoker fix. http://technet.microsoft.com/en-us/library/dd638389%28WS.10%29.aspx, June 2010. [Online; accessed 23-October-2011].

[41] Microsoft Corporation. Microsoft application virtualization 4.6 sp1 sequencing guide. http://download.microsoft.com/download/F/7/8/F784A197-73BE-48FF-83DA-4102C05A6D44/App-V/App-V%204.6%20Service%20Pack%201%20Sequencing%20Guide.docx, March 2011. [Online; accessed 16-October-2011].

[42] Microsoft Corporation. Prescriptive guidance for sequencing office 2010 using microsoft app-v 4.5 or 4.6. http://support.microsoft.com/kb/983462/en-us, October 2011. [Online; accessed 20-October-2011].

[43] Symantec Corporation. Symantec completes acquisition of altiris. http://www.symantec.com/about/news/release/article.jsp?prid=20070409_01, April 2007. [Online; accessed 5-November-2011].

[44] Symantec Corporation. Symantec workspace virtualization 6.1 sp6 user's guide. http://www.symantec.com/business/support/resources/sites/BUSINESS/content/live/DOCUMENTATION/3000/DOC3208/en_US/Symantec_Workspace_Virtualization_6_1_SP6_User%27s_Guide.pdf, 2010. [Online; accessed 5-November-2011].

[45] Internet Engineering Task Force. Real time streaming protocol (rtsp). http://tools.ietf.org/html/rfc2326, 1998. [Online; accessed 29-October-2011].

[46] Zentrum für Informationstechnologie im Bildungsbereich. Die münchner lern-it in harten zahlen. `http://www.zib.musin.de/zib-das-sind-wir/eckdaten/`. [Online; accessed 3-October-2011].

[47] Zentrum für Informationstechnologie im Bildungsbereich. Historisches. `http://www.zib.musin.de/zib-das-sind-wir/geschichte/`. [Online; accessed 3-October-2011].

[48] Tsveti Georgieva. Disadvantages of virtualization. `http://tsveti-georgieva.suite101.com/disadvantages-of-virtualization-a170745`, November 2009. [Online; accessed 26-October-2011].

[49] Scot Hillier. The system registry. `http://msdn.microsoft.com/en-us/library/ms970651.aspx`, 1996. [Online; accessed 01-November-2011].

[50] Galen Hunt and Doug Brubacher. Detours: binary interception of win32 functions. In *Proceedings of the 3rd conference on USENIX Windows NT Symposium - Volume 3*, pages 14–14, Berkeley, CA, USA, 1999. USENIX Association.

[51] Amir Husain. How to build an application virtualization framework. `http://vdiworks.com/wp/?p=15`, July 2008. [Online; accessed 10-October-2011].

[52] Citrix Systems, Inc. How application virtualization and session virtualization work. `http://www.citrix.com/English/ps2/products/subfeature.asp?contentID=1683975`. [Online; accessed 24-October-2011].

[53] Citrix Systems, Inc. Citrix application streaming guide. `http://support.citrix.com/article/CTX116414`, 2008. [Online; accessed 24-October-2011].

[54] Interra Information Technologies, Inc. Application virtualization - a whitepaper. `http://www.interrait.com/download/whitepapers/Whitepaper-ApplicationVirtualization.pdf`. [Online; accessed 26-October-2011].

[55] Softricity, Inc. Inside the grid. `http://www.softgridblog.com/storage/Inside%20the%20Grid%20Self%20Study.pdf`. [Online; accessed 14-October-2011].

[56] VMware, Inc. Vmware thinapp for application virtualization. `http://www.vmware.com/products/thinapp/overview.html`. [Online; accessed 27-October-2011].

[57] VMware, Inc. Introduction to vmware thinapp. `http://www.vmware.com/pdf/thinapp_intro.pdf`, 2008. [Online; accessed 27-October-2011].

[58] VMware, Inc. Vmware thinapp agentless application virtualization overview. `http://www.vmware.com/files/pdf/thinapp_intro_whitepaper.pdf`, 2008. [Online; accessed 23-October-2011].

[59] VMware, Inc. Vmware to expand desktop virtualization solution with acquisition of thinstall. `http://www.vmware.com/company/news/releases/thinstall.`

html, January 2008. [Online; accessed 23-October-2011].

[60] VMware, Inc. Vmware thinapp 4.6 – what's new? http://blogs.vmware.com/thinapp/2010/08/vmware-thinapp-46-whats-new.html, August 2010. [Online; accessed 5-November-2011].

[61] Chris Jackson. Making applications compatible with windows 7 in a virtualized environment. http://technet.microsoft.com/en-gb/magazine/ff458340.aspx. [Online; accessed 25-October-2011].

[62] Emre Kanlikilicer. What types of applications should i virtualize with server application virtualization? http://blogs.technet.com/b/serverappv/archive/2011/04/08/what-types-of-applications-should-i-virtualize-with-server-app-v.aspx. [Online; accessed 30-October-2011].

[63] Seung-Woo Kim. Intercepting system api calls. http://software.intel.com/en-us/articles/intercepting-system-api-calls/, August 2009. [Online; accessed 16-October-2011].

[64] Stefan Krempl and Edward Henning. Munich school network to be migrated to windows xp. http://h-online.com/-1195535, February 2011. [Online; accessed 3-October-2011].

[65] Oren Laadan and Jason Nieh. Operating system virtualization: practice and experience. In *Proceedings of the 3rd Annual Haifa Experimental Systems Conference*, SYSTOR '10, pages 17:1–17:12, New York, NY, USA, 2010. ACM.

[66] Tolly Enterprises, LLC. Tolly test report - application virtualization comparison: Vmware thinapp vs. microsoft app-v & citrix xenapp. http://www.vmware.com/files/pdf/TollyGroup-ThinApp-Test-Report-Aug09.pdf, August 2009. [Online; accessed 23-October-2011].

[67] Tim Mangan. Stories from the grid: osguard.cp. http://www.appvirtguru.com/viewtopic.php?f=7&t=2030, February 2007. [Online; accessed 14-October-2011].

[68] Tim Mangan. 10 years after the wow! (a look back at softricity by an insider, 10 years after their founding). http://www.brianmadden.com/blogs/timmangan/archive/2011/02/25/10-years-after-the-wow.aspx, February 2011. [Online; accessed 30-October-2011].

[69] Tim Mangan. I was a cloud service provider, and didn't know it! http://www.tmurgent.com/TmBlog/?p=608, October 2011. [Online; accessed 30-October-2011].

[70] Microsoft Developer Network. Hooks. http://msdn.microsoft.com/en-us/library/ms632589%28VS.85%29.aspx. [Online; accessed 10-October-2011].

[71] Mozilla Developer Network. Mcd, mission control desktop, aka autoconfig. https://developer.mozilla.org/en/MCD. [Online; accessed 09-November-2011].

[72] Joseph Nord. App streaming – kernel agents vs. all user mode. `http://blogs.citrix.com/2009/09/23/app-streaming-kernel-agents-vs-all-user-mode/`, September 2009. [Online; accessed 24-October-2011].

[73] Aaron Parker. Virtualizing adobe reader x. `http://blog.stealthpuppy.com/virtualisation/virtualising-adobe-reader-x/`, October 2010. [Online; accessed 19-October-2011].

[74] Michael Pietroforte. The advantages of application virtualization. `http://4sysops.com/archives/the-advantages-of-application-virtualization/`, July 2008. [Online; accessed 26-October-2011].

[75] Michael Pietroforte. The disadvantages of application virtualization. `http://4sysops.com/archives/the-disadvantages-of-application-virtualization/`, July 2008. [Online; accessed 26-October-2011].

[76] Microsoft Research. Detours. `http://research.microsoft.com/en-us/projects/detours/`. [Online; accessed 10-October-2011].

[77] Steven Roman. Win32 api programming with visual basic. `http://technet.microsoft.com/en-us/library/cc768129.aspx`, 2000. [Online; accessed 26-October-2011].

[78] Mark Russinovich. Inside the native api. `http://netcode.cz/img/83/nativeapi.html`, November 2004. [Online; accessed 26-October-2011].

[79] Mark Russinovich. Inside native applications. `http://technet.microsoft.com/en-us/sysinternals/bb897447.aspx`, November 2006. [Online; accessed 26-October-2011].

[80] John Sheehan. Architecture and engineering of microsoft application virtualization (appvirt). `http://channel9.msdn.com/Shows/Going+Deep/John-Sheehan-Application-Virtualization-Redux-Inside-AppVirt-45`. [Online; accessed 27-October-2011].

[81] John Sheehan. Virtualization. `http://channel9.msdn.com/Shows/Going+Deep/Virtualization`. [Online; accessed 27-October-2011].

[82] Amit Singh. An introduction to virtualization. `http://www.kernelthread.com/publications/virtualization/`, January 2004. [Online; accessed 18-October-2011].

[83] Ruben Spruijt. Application virtualization smackdown. `http://www.virtuall.nl/download-document/application-virtualization-smackdown`, 2011. [Online; accessed 26-October-2011].

[84] Credit Suisse. Desktop virtualization comes of age. `http://www.dabcc.com/documents/DesktopVirtualization_11_26_07.pdf`, November 2007. [Online; accessed 01-November-2011].

[85] Microsoft Office Team. Click-to-run: Delivering office in the 21st century. `http://blogs.technet.com/b/office2010/archive/2009/11/06/click-to-run-delivering-office-in-the-21st-century.aspx`, November 2009. [Online; accessed 12-October-2011].

[86] Microsoft VMM Team. Now available: The release candidate for system center virtual machine manager 2012. `http://blogs.technet.com/b/scvmm/archive/2011/09/08/now-available-the-release-candidate-for-system-center-virtual-machine-manager-2012.aspx`. [Online; accessed 30-October-2011].

[87] Jim Truchon. App-v and support for applications with custom shell extensions. `http://blogs.technet.com/b/virtualworld/archive/2009/12/09/app-v-and-support-for-applications-with-custom-shell-extensions.aspx`, 2009. [Online; accessed 28-October-2011].

[88] Franky Wong. Dll hell, the inside story. `http://desaware.com/tech/dllhell.aspxx`, 1998. [Online; accessed 01-November-2011].