Distributed Printing Services

Masterarbeit SS05

ETH Zürich

11. April – 11. Oktober 2005

Mischa Zehnder

(zehndemi@student.ethz.ch)

bei

Distributed Computing Group

Prof. Dr. R. Wattenhofer

betreut von

N. Burri

Inhaltsverzeichnis

INHALTSVERZEICHNIS 1

Danksagung 3

EINLEITUNG UND MOTIVATION 4

EIN KOMPLETTES DRUCKSYSTEM 5

STAND ZU BEGINN DER MA 6

ZIELE DER MA 6

DPS IN DER ÜBERSICHT 8

Aufzeichnung & Wiedergabe 9

Transport 10

DIE STRUKTUR VON DPS 11
- Der DPS Stamm 11
- Die Organisationseinheit 12
- Der Druckspeicher 13
- Die Warteschlange 14
- Der Drucker 14
- Navigieren und Identifizieren 15
- Jobs identifizieren 16

DPS AUFZEICHNUNG IM DETAIL 17

Einleitung 17

Windows 17
- Windows GDI Graphics Device Interface 17
- Drucken in Windows 18

Mit Windows im Netzwerk drucken 19

DPS Treiber **20**
Der DPS Druckertreiber für Windows 20
Dokumententreue 21
Doppelte Aufzeichnung? 23
GDI und DDI 23
Der DPS Druckerporttreiber für Windows 25

DPS WIEDERGABE IM DETAIL **27**
Das Druckmodul 27

DIE DPS SERVERDIENSTE **28**

Dienste hinter der Struktur **28**
Grafische Übersicht 29
Der Katalog-Dienst 29
Der Druckspeicher-Dienst 30
Der Warteschlangen-Dienst 31

Implementationsdetails **34**
Gegenseitiger Ausschluss 34
Die Garbage-Collection 35
Der Hintergrundprozess 36
Das Sicherheitssystem 36

DAS POSTSCRIPT-RELAIS **38**

ZUSAMMENFASSUNG **39**

WEITERE ARBEITEN **39**

Danksagung

Hiermit möchte ich allen Beteiligten dafür danken, dass sie es ermöglicht haben, dass DPS entstehen konnte. Dank dem grossen Freiraum, den ich bei der Erarbeitung dieser Masterarbeit hatte, konnte ich vielen Ideen nachgehen und viele Erfahrungen sammeln.

Einleitung und Motivation

Einen Druckauftrag in einem Netzwerk ausführen zu lassen ist heute nur eingeschränkt möglich. Aktuelle Lösungen, wie sie in aktuellen Betriebssystemen integriert sind, wurden fast ausschliesslich auf Firmen oder Heimnetzwerke ausgelegt, in denen jeder Benutzer seinen festen Arbeitsplatz hat und praktisch immer mit demselben Drucker arbeitet. Wenn ein Auftrag doch einmal an einen anderen als den Standarddrucker gesendet werden soll, wird erwartet, dass der Benutzer explizit eine Verbindung zu diesem Drucker herstellt und die Einstellungen für den Drucker von Hand vornimmt. Bei diesem Vorgang wird ein Treiber auf dem System installiert und der Benutzer muss sich mit den Eigenheiten des verwendeten Druckers vertraut machen.

Diese Art von Netzwerkdruck ist sehr schlecht für Druckdienste[1] geeignet, da hier der Benutzer nicht immer wissen kann, welche Drucker schlussendlich für den Druckauftrag verwendet werden und welchen Treiber er installieren soll. Zudem kann nicht gewährleistet werden, dass der Benutzer den richtigen Treiber verwendet, um die Druckdaten zu generieren. Dies kann zu fehlerhaften Ausdrucken oder im schlimmsten Fall sogar zum Absturz des Druckdienstes führen. Auch wenn der Treiber in einem System automatisch installiert und aktualisiert wird, hat es immer noch den Nachteil, dass der Client Einstellungen vornehmen muss, um dem Treiber mitzuteilen, welches Papier zum Beispiel momentan im Drucker eingespannt ist. Diese spezifischen Daten können nicht automatisch übermittelt werden. Oft sind solche Lösungen für den Benutzer zu kompliziert in der Anwendung.

An grösseren Institutionen wie der ETH, bei denen eine grosse Menge Benutzer an vielen verschiedenen Computern arbeiten können, ist es noch ungeeigneter, die aktuellen Systeme einzusetzen. An jedem Arbeitsplatz müsste sich der Benutzer wieder auf die Handhabung des gerade am nächsten stehenden Druckers einstellen, weil jeder einen anderen Treiber mitbringt. Wenn dann nicht jede Maschine jede Aufgabe gleich ausführt, kann das sehr ärgerlich und zeitaufwendig sein. Aus die-

[1] Zum Beispiel ein CopyShop, dem man Aufträge online schicken kann oder eine spezielle Abteilung in einer Firma, welche Grossformat- und Spezialdrucker betreibt.

sem Grund ist es wünschenswert, ein System zu haben, das alle Druckaufträge auf allen Druckern gleich ausführt, ohne dass sich für den Benutzer etwas ändert. Alle Druckaufträge müssen über eine einzige Schnittstelle konfiguriert und abwickelt werden können, egal wann, wo und mit welcher Maschine.

Ein komplettes Drucksystem

Neben den in der Einleitung genannten Faktoren, gibt es weitere Gründe, ein komplettes Drucksystem zu bauen und nicht eine kleinere Version, die einfach Druckaufträge umleitet.

Es gibt zwei Probleme, welche dazu geführt haben, das heute vorliegende System zu bauen. Das eine war das Problem, dass es keinen Druckdienst gibt, an den man Plakate senden kann, ohne grösseren Aufwand in Sachen Formatierung und Konvertierung zu betreiben. Es gibt zwar Dienste (wie z.B. VPP), welche, Druckaufträge ausführen, aber eben nicht so problemlos, wie man sich das wünscht. Es gibt keine Möglichkeit Vorschauen anzusehen, die Dateien sind unnötig gross und ein Transport zum Druckdienst ist relativ aufwendig. Die Dienste sind allesamt verbesserungswürdig.

Das zweite Problem stellen die Zusatzfunktionen dar, welche einige Treiber bieten, andere aber nicht. Der eine Hersteller bietet die Möglichkeit an, mehrere Seiten auf ein Blatt zu drucken, der andere nicht. Bei einem Modell das diese Funktion unterstützt, werden die Seiten genau so verkleinert, dass sie einem Bruchteil der Papiergrösse entsprechen. Bei anderen Modellen wird die Seite noch stärker verkleinert und eine Art Papierrand mit hinterlegtem Schatten aufgedruckt.

Es gibt viele Unterschiede bei diesen Zusatzfunktionen. Da sie aber alle durch Software simuliert werden können, ohne dass die Hardware eine spezielle Eigenschaft mitbringen muss, wurde DPS so ausgelegt, dass die Druckaufträge durch solche Funktionen verändert werden können.

Um die Druckaufträge genau so zu erstellen, dass die beiden genannten Probleme gelöst werden können, konnte nicht auf einen Standarddruckertreiber zurückgegriffen werden. Es musste ein spezielles Format für die Druckjobspeicherung geschaffen werden, weil kein Format, das in Windows verwendet wird, dies mit adäquatem Aufwand ermöglicht hätte. Zwar wäre es machbar, diese Aufgabe mit einem auf Postscript aufgebauten Treiber zu lösen. Da Postscript aber sehr komplex aufgebaut ist

und zur Interpretation fast eine eigene virtuelle Maschine benötigt, wurde die Lösung mit dem eigenen Treiber favorisiert.

Stand zu Beginn der MA

In einer vorangegangenen Semesterarbeit wurde bereits ein erster Teil von DPS implementiert. Natürlich fehlten viele Komponenten, die im endgültigen Produkt enthalten sein sollten. Auch wichtige Kernkomponenten waren nicht vollständig vorhanden oder zum Teil nicht einmal angedacht. Von einem „Programm“ konnte man noch nicht sprechen, bestenfalls von einer frühen Beta-Version.

So fehlte z.B. die ganze Druckerüberwachung. Diese ist notwendig, um einen Auftrag (und dessen Schriftarten), zum korrekten Zeitpunkt freizugeben. In der Lösung der Semesterarbeit wurde dies durch einen einfachen Timer gelöst. Es wurde davon ausgegangen, dass ein Auftrag nie länger als 60 Sekunden dauern würde. Natürlich war dies nicht wirklich eine Lösung. Dieses Thema wird in einem der folgenden Kapitel behandelt.

Zudem war am Ende der Semesterarbeit zu sehen, dass grosse Teile des Systems umgebaut werden mussten, um wirklich alle Anforderungen, die an das System gestellt wurden, erfüllen zu können.

Ziele der MA

Am Ende der Masterarbeit soll ein Produkt vorhanden sein, mit dem ein Druckdienst betrieben werden kann. Die bereits bestehenden Ansätze sollen soweit ausgebaut und komplettiert werden, dass ein vollwertiges Produkt entsteht.

Das System sollte resistent gegen diverse Ausfälle gemacht werden und dem Benutzer ein Feedback über den Status seiner Druckaufträge geben.

Neue Komponenten und Ideen wie die Kompression bei der Übertragung, Optimierungen und Druckfunktionen sollen hinzugefügt werden. Zudem soll ein Postcript-to-DPS Relais gebaut werden, das es ermöglichen soll, dass auch Unix-Benuzter an das System angeschlossen werden können.

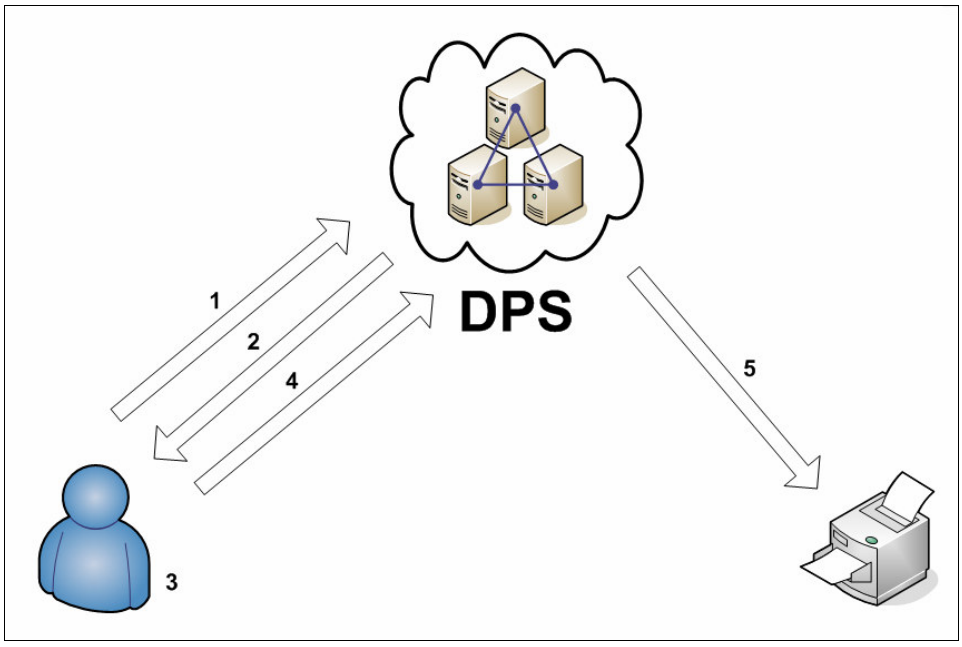
Das so entstandene Produkt soll dann für eine Testgruppe zugänglich gemacht werden um Erfahrungen für allfällige weiterführende Arbeiten oder als Vorbereitung für einen Einsatz in einem produktiven Umfeld sammeln zu können

DPS in der Übersicht

Als erstes wird in diesem Kapitel eine Übersicht über DPS und seine Grundfunktionen gegeben. Danach werden die einzelnen Abläufe detailliert beschrieben. In einem weiteren Kapitel werden dann die Komponenten genauer betrachtet.

DPS ist ein System, das Druckaufträge auf einem Clientcomputer aufzeichnet, sie dann auf einen Server überträgt und dort wiedergibt. Das System geht dabei in Schritten vor, die klar voneinander getrennt sind. Die Reihenfolge dieser Schritte ist in jedem Fall gleich.

Die folgende Grafik gibt einen groben Überblick über die einzelnen Schritte.



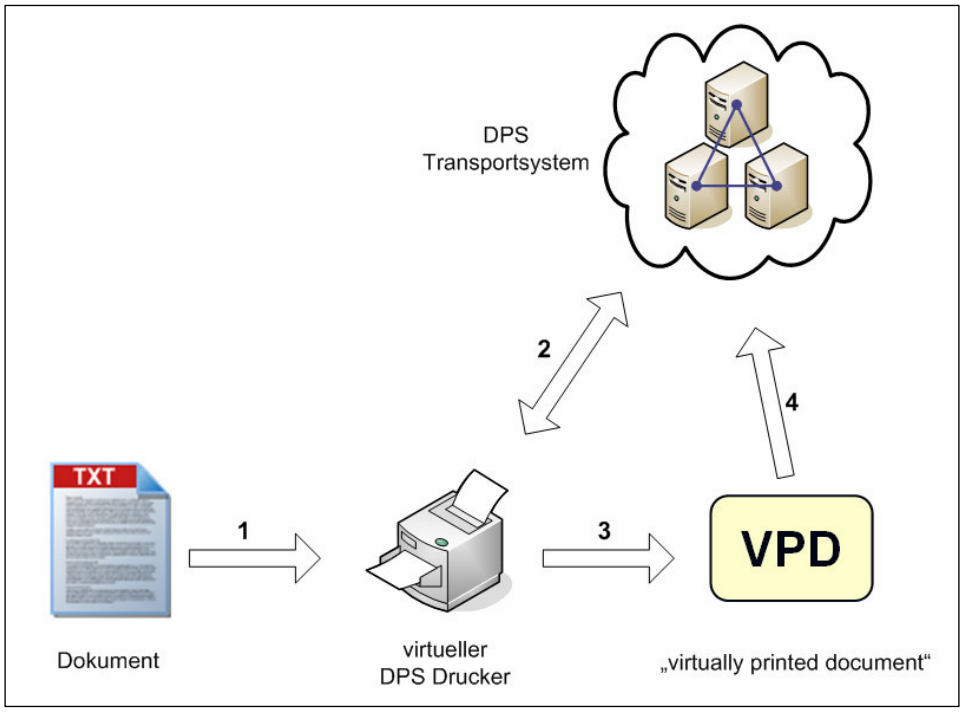
Im ersten Schritt fordert der Benutzer die benötigten Informationen zum Netzwerk an. Im zweiten Schritt sendet das System die Daten an den Benutzer zurück. Anschliessend zeichnet der Benutzer den Druckauftrag auf. Dann sendet er im vierten Schritt den Druckauftrag an DPS. Wurde der Auftrag vollständig übertragen, führt das System als fünften Schritt den Druckauftrag aus.

Im Folgenden wird die Aktion des Druckens zur besseren Verständlichkeit in zwei grundlegende Teile aufgeteilt. Der erste Teil ist für das Aufzeichnen und Wiedergeben zuständig (Schritt 3 und 5). Der zweite Teil regelt den ganzen Transport (Schritt 1, 2 und 4). Die beiden Teile sind völlig unabhängig voneinander.

Aufzeichnung & Wiedergabe

DPS verwendet zur Aufzeichnung des Druckauftrages einen eigenen Treiber. Auf dem Clientsystem wird mit seiner Hilfe ein virtueller Drucker simuliert, welcher die Schnittstelle zwischen Benutzer und DPS darstellt. Der DPS Drucker funktioniert wie jeder andere Drucker im System und ermöglicht daher einen sehr intuitiven Umgang mit dem Drucksystem.

Wie jeder Drucker bietet auch der DPS Drucker ein Konfigurationsmenü an. Wenn die Konfigurationsseite des Druckers aufgerufen wird, ruft der Treiber als erstes die Informationen zum verwendeten DPS Druckdienst ab. Die Informationen werden vom DPS Transportsystem bereitgestellt. Diese Daten über vorhandene Drucker und deren Eigenschaften verarbeitet der Treiber dann zu Konfigurationsseiten. Hat der Benutzer schliesslich den Zieldrucker ausgewählt und die gewünschten Optionen



eingestellt, kann er über die Anwendung den Druckauftrag starten.

Der Treiber zeichnet dann den Druckauftrag vollständig auf und übergibt ihn dem DPS Transportsystem.

Nachdem das Transportsystem den Druckauftrag zum Server transportiert hat, übergibt es ihn an das Druckmodul von DPS. Das Druckmodul ist das Gegenstück zum Druckertreiber. Aus den Daten, die der Druckertreiber aufgezeichnet hat, macht das Druckmodul wieder einen Druckauftrag. Ein DPS Druckauftrag endet, nachdem der Auftrag abgespielt und dem realen Drucker übergeben wurde.

Transport

In diesem Kapitel wird das Transportsystem von DPS genauer angesehen. Diese Komponente hat zwei Hauptaufgaben: Als erstes ist sie dafür verantwortlich, den Client mit den nötigen Informationen über die vorhandenen Drucker und Optionen des Druckdienstes zu versorgen (Schritt 1 und 2 der Übersichtsgrafik). Die zweite Aufgabe besteht im eigentlichen Transport der Druckdaten (Schritt 4 der Grafik).

Clients können die benötigten Informationen von DPS abrufen wie von einer grossen verteilten Datenbank. Der Vorgang, dass ein Client einen Druckauftrag an das System übermittelt, funktioniert wie das Schreiben von Daten in eine Datenbank.

Nachdem der Druckauftrag vom Treiber aufgezeichnet worden ist, generiert der Client diverse Einträge im DPS System. Wenn diese vollständig gemacht worden sind, beginnt das System mit der Abarbeitung des Auftrages.

Um einen Auftrag korrekt dem System zu übergeben, geht der Client in einer bestimmen Reihenfolge vor. Als erstes muss ein Speicherobjekt mit den eigentlichen Druckdaten angelegt werden. Danach werden die ganzen Daten des Druckauftrages in diesem Objekt abgelegt. Anschliessend wird einem weiteren Teil von DPS der Befehl erteilt, den Auftrag aus diesem Objekt auszuführen. Für diese verschiedenen Aufgaben sind verschiedene Elemente und Server zuständig, welche die eigentliche Struktur vom DPS Netzwerk ausmachen. Im nächsten Kapitel werden diese genauer erläutert.

Die Struktur von DPS

Das Transportsystem von DPS besteht, im Gegensatz zu den Aufzeichnenden und Wiedergebenden Modulen, nicht aus einem einzigen Teil, sondern gliedert sich in diverse Elemente. Diese Elemente bilden mit ihren Beziehungen untereinander die DPS-Netzwerk-Struktur. In diesem Kapitel wird diese Struktur genauer diskutiert.

Die DPS-Netzwerk-Struktur ist eine rein logische Struktur und hat keine physische Repräsentation. Sie besteht aus lose zusammenarbeitenden Servern. Das ganze DPS-Netz existiert nie als fix zusammenhängendes Netz. Konstante Verbindungen der einzelnen Server zueinander sind nie nötig. Der einzige Grund wieso sie in der aktuellen Version dennoch bestehen ist die Performance. Dauerndes Auf- und Abbauen von Verbindungen ist teuer, weshalb häufig verwendete Verbindungen konstant aufrechterhalten werden.

In der Struktur gibt es Objekte, die als eines oder mehrere Serverobjekte instanziert werden. Dazu gehören der DPS Stamm, der Druckspeicher und die Warteschlange. Andere Objekte, namentlich die Organisationseinheit und das Druckerobjekt, sind rein virtuelle Objekte, ohne eine Serverinstanz.

Im Folgenden werden die 5 Elemente und deren Funktionen, die eine DPS Struktur ausmachen, genauer beschrieben. Das System orientiert sich an den heute üblichen Baumstrukturen, wie sie in Dateisystemen und heutigen Domänenverwaltungen verwendet werden.

In einem nächsten Kapitel werden die Dienste erläutert, welche die Serverinstanzen der hier vorgestellten logischen Elemente ausführen, steuern und überwachen.

Der DPS Stamm

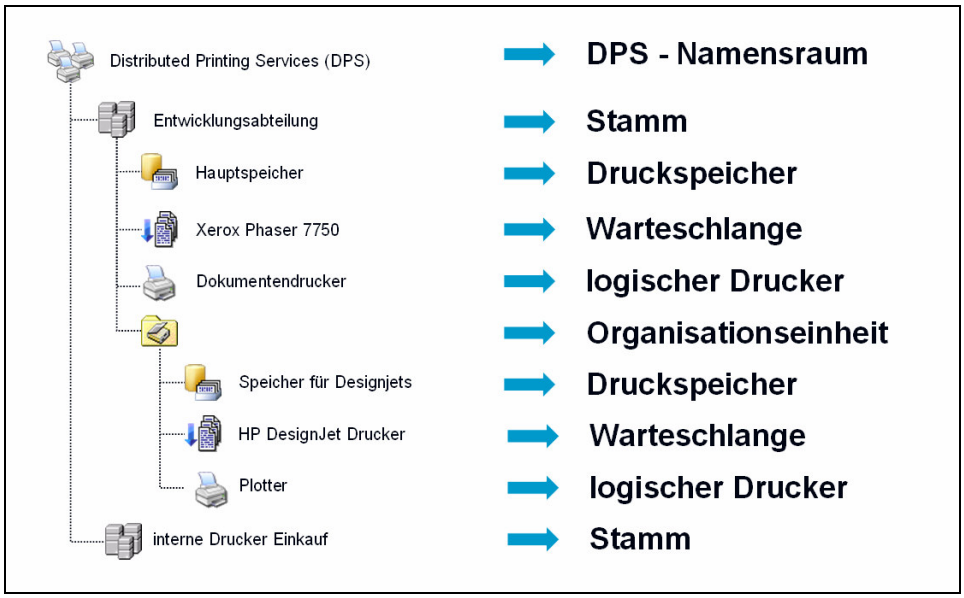
Eine DPS-Struktur besteht immer aus einem oder mehreren DPS-Stämmen. Ein Stamm ist der oberste Knoten von DPS und ist mit einer Domäne zu vergleichen. Der Stamm bildet einen abgeschlossenen Na-

mensraum. Alle Objekte, die zu einem Druckdienst gehört, werden in einem Stamm zusammengefasst.

Der Stamm hat mehrere Funktionen: Zum einen dient er als Katalog für alle DPS-Objekte. Wenn ein Client eine Verbindung zu den DPS-Diensten herstellt, muss er zuerst eine Verbindung zum DPS-Stamm herstellen. Nur so kann er die Unterobjekte finden und auf diese zugreifen.

Der Stamm stellt zudem den obersten Sicherheits-Container dar. Er definiert, welchen Benutzern der Zugriff auf seine Unterobjekte gewährt und welchen Personen er verweigert wird. Der Stamm funktioniert bezüglich Sicherheit wie eine ActiveDirectory Domäne, eines Windows Servers.

In der untenstehenden Grafik ist ein Beispiel eines DPS-Namensraumes zu sehen.



Die Organisationseinheit

Die Organisationseinheit von DPS ist ein Instrument um Objekte zusammenzufassen. Wie der Stamm auch, kann eine Organisationseinheit Druckspeicher, Warteschlangen, Drucker und Organisationseinheiten enthalten. Sie funktionieren genau wie der Stamm als ein Sicherheits-Container. Man könnte sie als eine Art Leichtgewicht-Stamm bezeichnen. Um Objekte zu gruppieren und sie für verschiedene Benutzer freizugeben, bieten sie ein gutes Hilfsmittel.

Eine Organisationseinheit ist jedoch lediglich ein administratives Element. Ein Benutzer des Systems wird nie eine Organisationseinheit sehen. Für ihn erscheint der Namensraum flach, also unstrukturiert, wie ein einzelner Behälter, in dem alle Objekte enthalten sind.

Die Organisationseinheiten sind für die einzelnen Serverobjekte jedoch sehr wichtig. Von ihnen erhalten sie nämlich die ganzen Sicherheitseinstellungen, wenn sie sich beim Stamm anmelden.

Im Gegensatz zum Stamm hat die Organisationseinheit keine Repräsentation als Serverobjekt. Sie ist eines der beiden rein virtuellen Objekte.

Der Druckspeicher

Um einen Auftrag ausführen zu können, benötigt das Drucksystem den Druckauftrag. Da die Übermittlung der Daten rein statistisch gesehen eher als jede andere Kommunikationsphase von Client zu Server unterbrochen werden kann, wurde sie von den anderen Aufgaben getrennt. (Die Übertragung besteht aus der Übermittlung von etlichen tausend Mal mehr Daten, als die ganzen Steuersignale, welche der Client an den Server sendet, um den Druckauftrag zu platzieren. Daher ist es am wahrscheinlichsten, dass ein Fehler in dieser Phase und nicht in einer anderen passiert).

Durch diese Trennung wird verhindert, dass das ausführende System jemals etwas von einem unvollständig übermittelten Auftrag erfährt. Es kann auch nicht passieren, dass nur 10 Seiten von 15 übermittelt werden und der Druck mitten in der Ausführung stecken bleibt.

Soll ein Auftrag ausgeführt werden, muss also zuerst der ganze Datensatz auf den passenden Druckspeicherserver geladen werden. Anschliessend kann dieses Objekt bei einer Warteschlange angemeldet werden, welche dann die Aufgabe ausführt und von nun an die Verwaltung des Datenobjektes übernimmt.

Ein Druckspeicher wird vom DPS-Speicher-Dienst (Storage Service) verwaltet. Der Druckspeicher ist das eigentliche Serverobjekt, das vom Speicher-Dienst ausführt und überwacht wird.

Die Warteschlange

Die Warteschlange ist das wichtigste Objekt von DPS. Der DPS Benutzer kann einen Auftrag in einer Warteschlange erzeugen, sobald er seinen Druckjob auf einen Druckspeicher geladen hat. Um dies zu tun, übergibt er die ID des Datenobjekts an die Warteschlange. Die Warteschlange sucht dann das Datenobjekt. Alle weiteren Informationen über den Auftrag, wie z.B. die Anzahl Seiten die zu Drucken sind, die einzelnen Optionen, die Blattgrösse oder die Qualität, sind in diesen Daten gespeichert und müssen daher nicht separat an die Warteschlange übergeben werden.

Die Warteschlange bildet eine Schnittmenge der Eigenschaften der lokalen Drucker, die sie verwendet und sie publiziert diese Daten im Stamm, wo sie die Clients abrufen können. Wenn ein Auftrag ausgeführt werden soll, ist es die Warteschlange, die entscheidet, auf welchem Drucker die Ausgabe tatsächlich gemacht wird.

Neben dieser Verteilaufgabe ist das Objekt auch dafür verantwortlich den Status der angeschlossenen Drucker zu überwachen und allfällige Probleme an eine zentrale Stelle zu melden. Hat z.B. ein Drucker kein Papier mehr, dann teilt dies die Warteschlange dem Stamm mit und ein Administrator kann zentral alle Fehlermeldungen gesammelt abrufen. Dadurch ist es möglich, das ganze System von einem einzigen Arbeitsplatz aus zu überwachen.

Der Drucker

Das Gerät, um welches sich alles dreht in DPS, ist der Drucker. Das logische Element „Drucker“ hat in DPS zwar auch eine zentrale Rolle, es fungiert jedoch nur als eine Verknüpfung im Namensraum.

Ein Drucker verknüpft einen oder mehrere Druckspeicher und Warteschlangen mit einem Namen. Sein einziger Zweck ist es, dem Client zu ermöglichen, den richtigen Server zum gewünschten Druckernamen zu finden.

Sind mit dem Drucker mehrere Warteschlangen oder Druckspeicher verknüpft, dann wählt der Katalog (der Stamm) anhand eines Rundlaufverfahrens aus, an welches Objekt er den anfragenden Client verweist. Damit kann die Last auf verschiedene Computer aufgeteilt werden.

Das Druckerobjekt hat mit dem realen Drucker nicht viel zu tun. Über welchen Drucker das System schlussendlich wirklich druckt, hängt von

den Einstellungen der Warteschlange und des Warteschlangendienstes ab. Die eigentliche Maschine, welche den Druck ausführt, kann also von Druckjob zu Druckjob unterschiedlich sein, obwohl der Benutzer immer denselben Drucker auswählt.

So kann zum Beispiel ein „Urkundendrucker“ definiert werden, der heute über einen „HP LaserJet III“ druckt, der am Server WS1 angehängt ist. Später könnte dann dieser Drucker durch einen „HP LaserJet 4200“ ersetzt werden, der über den Server WS2 ansprechbar ist. Von all dem würde der Benutzer von DPS aber nichts merken. Für ihn bleibt alles beim Alten. Er druckt über den „Urkunendrucker“.

Navigieren und Identifizieren

In einem DPS-Netzwerk gibt es, wie in den letzten Kapiteln beschrieben wurde, mehrere verschiedene Objekte, die jeweils auf einem oder mehreren Servern instanziert sind. Um in einem solchen Netzwerk kommunizieren zu können, muss jedes dieser Objekte und jede dieser Serverinstanzen einen eindeutigen Namen haben.

Nummern eignen sich dazu nicht besonders gut. Bei den logischen Objekten wäre der Ansatz zwar noch praktikabel, da diese zentral registriert werden. Spätestens die einzelnen Serverinstanzen könnten aber nicht nummeriert werden, weil sie zu einem Zeitpunkt aufgebaut werden, in dem keine Verbindung zu einer zentralen Stelle existiert. Ausserdem sollten die Objekte ihren Namen nie ändern, da sonst die darauf erstellten Aufträge und Speicherobjekte nicht mehr gefunden werden können (mehr dazu unter *„Jobs identifizieren“*). Unter den Bedingungen wäre der Aufwand einfach zu gross, um das ganze zu koordinieren.

Die Lösung für das Problem sind so genante Globaly Unique Identifyers (GUIDs). In Windows werden diese speziellen Nummer- und Zahlenfolgen oft verwendet, wenn ein Objekt einen eindeutigen Namen erhalten soll. Die Formel, welche zur Bildung von GUIDs verwendet wird, garantiert dabei, dass die erhaltenen IDs tatsächlich weltweit einzigartig sind.

Jedes logische Objekt in DPS besitzt eine solche GUID als Identifikation. Zusätzlich wird auch jeder Dienst mit einer GUID identifiziert. Die Serverinstanz eines logischen Objekts auf einem Dienst, wird dann mit zwei GUIDs bezeichnet. Die GUID des logischen Objekts wird dazu an die GUID des Dienstes angehängt.

Wenn ein Client oder ein Server eine bestimmte Instanz eines Objektes sucht (z.B. wenn die Warteschlange eine Druckspeicher-Instanz sucht, um auf das benötigte Datenelement zugreifen zu können), kann sie an-

hand der GUID beim Katalog (dem Stamm) die aktuelle Adresse des Servers erfahren.

Auch logische Drucker werden über GUIDs identifiziert. Der zugewiesene Name ist dabei nicht wichtig. So kann der „Urkundendrucker“ aus dem letzten Kapitel theoretisch auch in „neuer Urkundendrucker“ umbenannt werden. Der Druckertreiber auf dem Client-Computer erkennt automatisch, dass sich nur der Name geändert hat und zeigt dies so an.

Jobs identifizieren

Wenn ein Benutzer in einem Druckspeicher ein Objekt erstellt oder einen Druckjob generiert, wählt er nicht selbest aus, mit welchem Computer er kommuniziert. Der DPS-Katalog verweist ihn an einen beliebigen Computer, der eine Instanz des gewünschten Objekts besitzt. Um den heraufgeladenen Druckauftrag in einer Warteschlange zu platzieren, später den Status abzurufen oder den Job zu löschen, muss jedes Objekt (Datenobjekte im Druckspeicher und Auftragsobjekte in Warteschlangen) eindeutig identifizierbar sein. Darum generiert DPS für jedes Element eine global eindeutige ID. Diese besteht immer aus der ID der Server-Instanz, die das Objekt erstellt hat, gefolgt von einem Zeitstempel und einer fortlaufenden Zahl modulo 2^32. Mit dieser ID lässt sich ein Objekt immer zweifelsfrei identifizieren, selbst wenn man es mit Objekten eines fremden Stammes vergleichen würde, was aber nie vorkommt.

Wird ein Objekt gesucht, kann der Client den erstellenden Server kontaktieren (seine ID ist ja in der ID des Objekts enthalten). Sollte das Objekt aus irgendeinem Grund verschoben worden sein und jetzt auf einem anderen Server liegen, weiss der erstellende Server immer wo das Objekt jetzt ist. Es sollte jedoch selten vorkommen, dass ein Objekt verschoben wird. Daher kann man davon ausgehen, dass in der ObjektID jeweils die Adresse des Servers enthalten ist, auf dem das Objekt momentan gespeichert ist.

DPS Aufzeichnung im Detail

Einleitung

Bevor man den Aufzeichnungsvorgang von DPS im Detail verstehen kann, muss man das Drucksystem von Windows verstehen, auf dem der virtuelle Drucker von DPS aufbaut. Die Idee von Windows ist es, dass die Anwendung welche eine Ausgabe auf dem Drucker macht, nicht wissen muss mit welchem Drucker sie wirklich arbeitet und wie er funktioniert. Die Steuerung des Druckers wird an einen Treiber übertragen.

Windows

Windows GDI
Graphics Device Interface

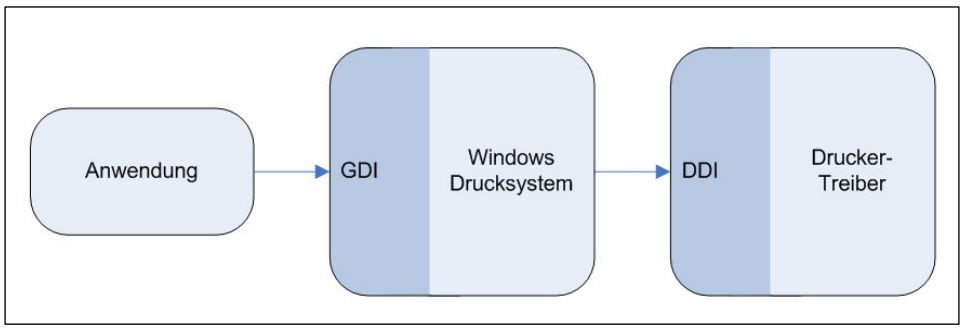
Das Graphical Device Interface ist die gemeinsame Schnittstelle für alle Zeichengeräte in Windows. Sie ist das Bindeglied zwischen der Anwendung und dem Ausgabegerät. Diese Ausgabegeräte können sehr verschieden sein. Z.B. ein Bildschirm, ein Tintenstrahldrucker, ein Laserdrucker oder exotische Maschinen wie eine Stickmaschine, die automatisch Bilder aufstickt.

Der Vorteil an diesem System ist, dass die Anwendung keinen speziellen Code mitbringen muss, um die Bildschirmausgabe auf den Drucker zu bringen. Alles was sie tun muss, ist Windows mitzuteilen, dass die Daten jetzt nicht auf dem Bildschirm erscheine, sondern an den Drucker gesendet werden sollen.

Um den Applikationsentwickler, wie auch den Treiberentwickler zu entlasten, besitzt GDI eine leicht andere Schnittstelle als die Treiber für die

Ausgabegeräte. GDI führt gewisse transformierende Funktionen durch und liefert dem Treiber vereinfachte oder zusätzliche Informationen, welche er für die Ausführung benötigt. Eine Hauptaufgabe von GDI besteht z.B. im zwischenspeichern von verwendeten Daten, welche die Anwendung sonst bei jedem Aufruf neu senden müsste.

Die nachfolgende Grafik zeigt eine Übersicht über den Aufbau des Drucksystems von Windows. Es setzt die Aufrufe an die GDI Schnittstelle (Graphics Device Interface) um und macht Aufrufe an die DDI Schnittstelle (Device Driver Interface) des Treibers.



Drucken in Windows

Wenn eine Anwendung einen Drucker als Ausgabegerät ausgewählt hat, dann lädt Windows den entsprechenden Treiber und übermittelt ihm die Befehle der Anwendung. Der Treiber wandelt diese dann um und sendet sie an den Drucker weiter. So sieht das zumindest aus der Sicht des Benutzers aus, wenn er einen Druckauftrag gibt.

Drucker sind jedoch nicht sehr schnell. Während der Bildaufbau am Bildschirm nur wenige Millisekunden dauert, braucht der Drucker schon 10 Sekunden, bis er überhaupt anfängt ein Papier aus dem Schacht einzuziehen, und das auch nur bei einem sehr schnellen Drucker.

Wenn der Druck begonnen hat, kann es noch einmal über eine Minute dauern, bis die Seite bedruckt worden ist. Bei Grossformatdruckern, welche Plakate drucken, dauert dieser Vorgang sogar noch länger. Manchmal einige Stunden oder im Extremfall sogar Tage. Würde das Drucksystem synchron arbeiten (blockieren, bis die Ausgabe gemacht wurde), wie es die Bildschirmausgabe tut, dann wäre die Anwendung im schlimmsten Fall eine lange Zeit komplett blockiert. Man müsste warten, bis der Druck abgeschlossen ist, bevor der Benutzer weiterarbeiten könnte.

Um diese Probleme zu lösen, kommt bei Windows der Spooler zum Einsatz. Er macht nichts anderes, als die Zeichenoperationen der Anwendung aufzuzeichnen und sie danach etwas langsamer auf dem Drucker

wiederzugeben. Dadurch kann die Anwendung die Ausgabe schnell beenden und der Benutzer kann weiterarbeiten.

Der Spooler arbeitet danach den Druckauftrag in zwei Stufen ab. Als erstes transformiert er die aufgezeichneten GDI Aufrufe der Anwendung in entsprechende DDI Aufrufe und macht dann die Aufrufe an den Druckertreiber. Jeder Aufruf wird vom Treiber direkt in einen Datenstrom aus speziellen Druckerbefehlen umgewandelt und an den Druckport gesendet. Hier ist der Spooler nochmals dazwischengeschaltet. Weil das Senden der Daten an den Drucker über den Port ebenfalls zu lange dauert und den Druckprozess nur unnötig belasten würde, puffert der Spooler auch diesen Datenstrom.

In einer zweiten Stufe sendet der Spooler die Daten, welche er vom Druckertreiber zurückerhalten hat, an den Drucker. Erst jetzt, möglicherweise Stunden nachdem die Anwendung den Auftrag erstellt hat, kommen die Daten beim Drucker an.

Mit Windows im Netzwerk drucken

Neben den erläuterten Verfahren um Druckaufträge lokal zu verarbeiten, bietet Windows die Möglichkeit, einen Auftrag per Netzwerk an einen anderen Computer zu senden. Windows führt dann den Auftrag auf dem Zielcomputer aus, an dem der eigentliche Drucker angeschlossen ist. Dies wird durch die bereits vorhandene Struktur des Drucksystems begünstigt. Die aufgezeichneten Druckjobs werden nicht lokal verarbeitet, sondern zuerst über das Netzwerk an den anderen Computer gesendet (inklusive aller benötigten Daten wie gewünschte Einstellungen des Druckers, Schriftarten etc.) Erst danach werden die zwei Schritte der Ausführung auf dem Servercomputer durchgeführt.[2]

Diese Aufgabenteilung hat Vor- und Nachteile. Einer der grössten Nachteile ist, dass der Clientcomputer denselben oder zumindest einen kompatiblen Treiber installiert haben muss, wie der Server. Diese Ab-

[2] Windows Vista (Codename Longhorn) wird diese Aufgaben etwas anders lösen. Wenn der neu eingeführte Metro-Druckpfad verwendet wird, wird der erste Schritt auf dem Client Computer ausgeführt und erst dann werden die Daten gesendet. Da es sich beim Metro-Druckpfad aber um ein anderes Verfahren handelt, das von andern Voraussetzungen ausgeht, lässt es sich schlecht mit dem GDI-Druckpfad vergleichen. Zudem werden in Windows Vista nicht alle Druckaufträge mit diesem System bearbeitet. Der bisher genutze GDI-Druckpfad wird weiterhin bestehen und dieser soll nach aktuellen Informationen nicht verändert werden.

hängigkeit vom richtigen Treiber macht das System schwer nutzbar, wenn es in grossen Netzwerken mit vielen verschiedenen Druckern eingesetzt werden soll, bei denen jeder Drucker einen anderen Treiber verwendet. Die Clients haben es dann mit einer Flut von Treibern zu tun und die Überwachung und Wartung einer solchen Struktur wird sehr aufwendig.

DPS Treiber

Der DPS Druckertreiber für Windows

Die Druckaufträge müssen auf der Maschine des Clients aufgezeichnet werden, damit sie danach an den Druckserver weitergeleitet werden können. Diese Aufgabe übernimmt der DPS-Druckertreiber. Der Druckertreiber auf dem Client System ist eines der komplexesten Teile an DPS. Er simuliert auf dem Computer einen virtuellen Drucker, den Programmen wie jeden andere Drucker verwenden können. Im Unterschied zu einem normalen Druckrtreiber wandelt er den Auftrag aber nicht in eine Druckersprache um, sondern in das spezielle VPD Format. VPD steht für „virtually printed document“. Dieses spezielle Format enthält alle Informationen, die nötig sind, um den Druckauftrag später auf dem Server exakt so durchzuführen, wie er ursprünglich von der Anwendung an den Drucker gesendet worden ist. Weil der Treiber ein gänzlich neues Format verwendet, lassen sich keine Standardkomponenten von Windows oder Vorlagen für Druckertreiber verwenden. Es liess sich daher kein Miniport-Treiber entwickeln, der sich dadurch ausgezeichnet hätte, dass er nur die Teile hätte implementieren müssen, die an diesem Drucker nicht einem Standarddrucker entsprechen. Die Entwicklung wurde dadurch erheblich komplexer, weil ein kompletter Treiber erstellt werden musste. Dieser Umstand erlaubte dafür im Gegenzug, dass die Möglichkeit bestand, Komponenten und Verhalten genau nach Wunsch anzupassen und zu implementieren.

Der Treiber ist neben der Übersetzungsaufgabe (dem Rendering) auch dafür verantwortlich, alle benötigten Eigenschafts- und Einstellungsfenster zu generieren, um den Drucker und die Ausgabeoptionen zu konfigurieren. Diese Optionen müssen vom Treiber so zusammengestellt werden, dass sie auch übermittelt werden können.

Der virtuelle Drucker von DPS stellt an dieses Konfigurationsmenü ganz spezielle Anforderungen. Er verhält sich komplett anders als ein normaler Treiber. Auch hier war es nötig, Techniken einzusetzen, die wahrscheinlich noch nie in einem Druckertreiber verwendet worden sind. Da der Treiber im Voraus keine Informationen über die Fähigkeiten des Druckers kennt und sie per Netzwerk abrufen muss, konnten viele Funktionen nicht statisch definiert werden. Zudem verändern sich die Eigenschaften sogar während der Konfiguration (z.B. ändert die maximale Papiergrösse durch das Auswählen eines anderen Zieldruckers). Der Treiber muss viele Informationen online generieren, was in normalen Treibern nicht oder nur in begrenztem Rahmen der Fall ist.

Allein der Umstand, dass das Druckerkonfigurationsprogramm des Treibers kommunizieren können muss, führte teilweise zu grossen Problemen, weil Windows nicht dafür ausgelegt ist.

Dokumententreue

Ein grosses Problem bei heutigen Druckdiensten und anderen Lösungen zur Druckumleitung sind die Veränderungen, welche das System im Dokument verursacht, bis es endlich am Drucker ankommt. Es ist natürlich sehr ärgerlich, wenn ein Dokument durch das System nicht so ausgedruckt wird, wie es auf dem Bildschirm erscheint, nachdem man es stundenlang designt hat. Schuld daran ist in den wenigsten Fällen die Software, mit der das Dokument erstellt wurde. Diese sendet dieselben Zeichenbefehle an den Drucker, wie bei der Vorschau an den Bildschirm. Die ungenaue Wiedergabe muss also beim Aufzeichnen, dem Transport oder der Wiedergabe passieren.

Viele Lösungen zur Druckumleitung „vergessen“ bei der Aufzeichnung der Aufträge, die Schriftarten zu erfassen und in die Übermittlung einzubeziehen. Bei einigen Systemen sind die Optionen zwar vorhanden, müssen aber explizit über Optionsschalter, mit zum Teil für den Laien nichts sagenden Namen wie „TrueType Fonts als Softfont laden“, aktiviert werden. Eine weit verbreitete Vorgehensweise ist auch, dass Bilddaten in JPEGs umgewandelt werden, bevor sie gesendet werden. Dies spart zwar Platz bei der Übertragung, bei einigen Bildern kann dadurch aber ein Kästchen-Effekt auftreten, wenn die Kompressionsrate für dieses spezielle Bild zu gross gewählt wurde. In so einem Fall hat man dann meist keine andere Möglichkeit, als einen anderen Druckdienst zu verwenden.

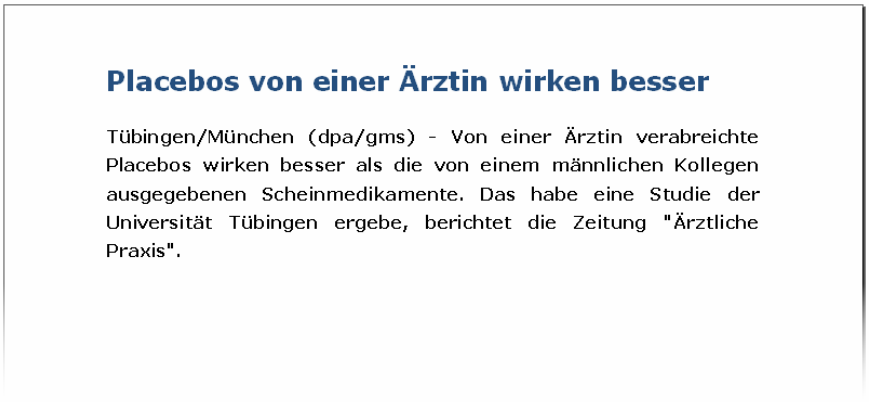
Ein weiterer möglicher Fehler, den man häufig antrifft, sind verzerrte Ausgaben. Wenn das aufzeichnende System z.B. von einem A4 Papier

ausgeht, das ausführende System dann aber den Auftrag auf das Letter-Format skaliert, können zum Teil seltsame und unerklärliche Schriftbilder entstehen. Ein ebenfalls häufig beobachtetes Problem ist die Skalierung anhand der Druckränder. Dabei wird das Blatt so weit verkleinert, dass es komplett in die bedruckbare Fläche des verwendeten Druckers passt. Dies hat dann meist ein Skalierungsfaktor von etwa 0.95 zur Folge.

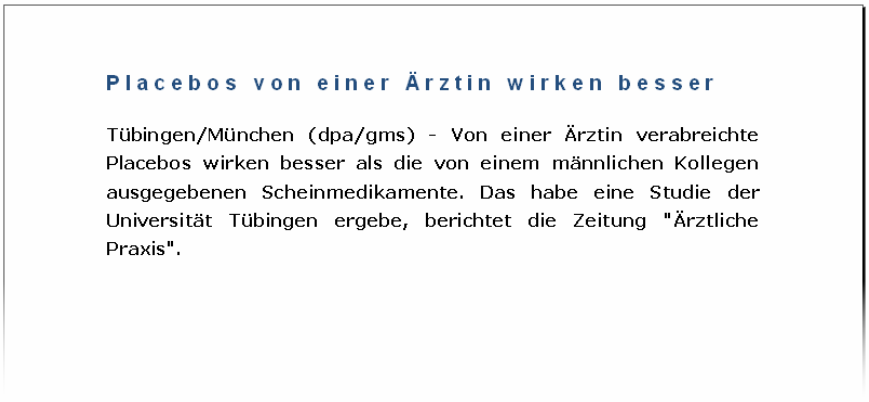
Jeder dieser Fehler ist sehr störend und oft braucht es grosse Anstrengungen und sehr viel Zeit um diese zu umgehen. Deshalb sollten sie auf jeden Fall vermieden werden.

Bei der Entwicklung von DPS wurde sehr viel Wert darauf gelegt, dass die Ausführung des Auftrages in keiner Weise durch das Drucksystem verändert wird. Bei der Geschwindigkeit heutiger Netzwerke ist es auch nicht mehr so wichtig, bei der Übertragung so wenige Daten wie möglich zu senden. Im Zeitalter der Gigabit-Netzwerke kann ein Auftrag durchaus 4 kB grösser sein, wenn man dafür die Qualität erhält, die man sich vorgestellt hat.

Folgende Abbildung zeigt als Beispiel ein Dokument, wie es auf dem Bildschirm erscheint und an einem lokalen Drucker ausgedruckt wird.



Dieses Bild zeigt die tatsächliche Ausgabe, wenn das verarbeitende System die verwendete Schriftart nicht korrekt verarbeitet.



Doppelte Aufzeichnung?

Es fällt auf, dass es beim Druckvorgang zwei verschiedene Aufzeichnungen gib. Zum einen zeichnet der Spooler den Druckauftrag auf und gibt ihn dann stückchenweise an den Druckertreiber weiter, zum anderen zeichnet der spezielle DPS Druckertreiber den Auftrag auf, indem er den VPD Datenstrom generiert.

Es ist durch die Struktur der Spoolerdaten nicht möglich, eine andere Lösung zu verwenden. Zwar liessen sich mit den richtigen Treibern die Spoolerdaten abfangen (dies ist in Windows sogar vorgesehen), das Problem dabei ist aber, dass das Format der Daten, die der Spooler erzeugt, nicht dokumentiert ist. Es gibt zwar einige Programme, welche versuchen, dieses Format zu interpretieren, gewisse Aufträge enden aber mit einer Fehlermeldung, da der Spooler einem unbekannten Befehl in die Daten eingefügt hat. Zudem wäre es auch nicht möglich, diese Daten einfach blind von einem Drucker auf einen anderen zu übertragen. Es ist leicht zu sehen, was passieren würde, wenn der Befehl, zum Setzen des Koordinatensystems, von einem Drucker mit A4 Papier 1:1 auf einen Drucker mit einer Papierrolle übertragen würde. Das ganze Bild würde skaliert. Statt der normalen Breite von 21 cm würde das Dokument jetzt mit einer Breite der Rolle ausgedruckt. Das könnte je nach Modell 1½ m oder sogar noch mehr sein.

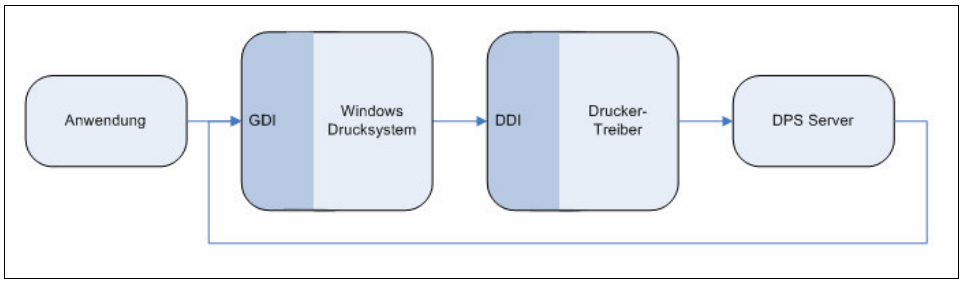
Weiter ist das Spoolerformat nicht so aufgebaut, dass es z.B. Schriftarten enthält (trotz der fehlenden Dokumentation lies sich das durch einige Versuche herausfinden). Druckdaten im Spooler sind daher nicht maschinenunabhängig. Sobald die Maschine, welche die Daten erzeugt hat, nicht mehr erreichbar ist, kann es sein, dass wichtige Daten fehlen (vor allem Schriftarten).

Aufgrund dieser Probleme gab es für DPS nur die Möglichkeit, einen eigenen Treiber zu verwenden, der die Druckbefehle passend aufzeichnet.

GDI und DDI

GDI (Graphics Device Interface), die Schnittstelle für Anwendungen, bietet einen gewissen Umfang an Funktionen, welche verwendet werden müssen, wenn ein Dokument an einen Drucker gesendet werden soll. DDI (Device Driver Interface) hingegen definiert die Schnittstelle, welche die Druckertreiber implementieren müssen.

Die Funktionen von GDI und DDI sind ähnlich, stimmen aber nicht ganz überein. Eine 1:1 Aufzeichnung ist daher nicht möglich, denn die Aufzeichnung muss Daten erzeugen, welche später wieder an GDI übergeben werden können (siehe Grafik).



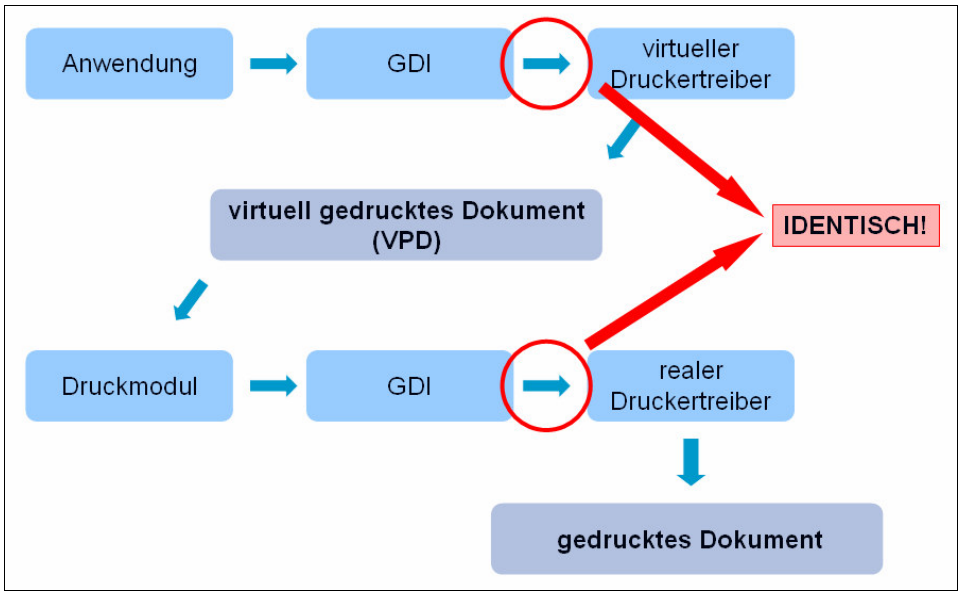
Eine besondere Schwierigkeit besteht darin, dass die DDI Schnittstelle so ausgelegt wurde, dass sie den Geräten immer genau die Daten übergibt, welche sie gerade brauchen um den Auftrag auszuführen. Dabei muss sich der Treiber nicht um das Cachen von Schriftarten kümmern. Er muss auch keine Clipping-Region verwalten oder sonstige Objekte zwischenspeichern. Allgemein muss er nur wenige Daten wirklich im Speicher halten. Alles wird immer wieder neu über die DDI Schnittstelle zur Verfügung gestellt. Würde der DPS-Druckertreiber diese ganzen Parameter jedes Mal neu in den Datenstrom integrieren, würde dieser viel zu gross. Der Treiber muss daher eine Art Online-Parsing der Daten durchführen und interpretieren was er braucht und was er schon von früher kennt. Da GDI z.B. die Schriftarten und die Clipping-Region zwischenspeichert, DDI diese aber jedes Mal neu sendet, können hier pro Befehl mehrere Kilobytes im VPD Datenstrom eingespart werden.

Anhand dieser Interpretation der Daten muss der Treiber einen Datenstrom, bestehend aus speziellen DPS-Steuersignalen und GDI-Aufrufen generieren und ausgeben. Da GDI den Anwendungen das Zwischenspeichern von Zeichenobjekten zur Verfügung stellt, sollte der Treiber versuchen, die Aufgaben so anzuordnen, dass diese Eigenschaft optimal ausgenutzt werden kann.

Weil das Drucksystem von Windows als eine Art Einwegfunktion betrachtet werden kann, ist die Aufgabe nicht immer einfach. Mehrere diverse GDI Aufrufe können zu denselben DDI Aufrufen führen. Je nach Parameter kann ein GDI Aufruf auch zu mehr als einem DDI Aufruf oder ein GDI Aufruf zu mehr als einem DDI Aufruf führen, da das Drucksystem keinen Einfluss auf die Ausgabe haben soll, muss der Treiber versuchen, aus den DDI Aufrufen diese GDI Aufrufe zu generieren, die später wieder zu denselben DDI Aufrufen führen. Aufgrund der Einwegfunktionseigenschaft von GDI ist es unmöglich, die Aufrufe der Anwendung zu erraten. Dies ist aber nicht nötig, da schlussendlich soll ja nur der Zieltreiber ex-

akt gleich angesteuert werden soll. Dazu muss der Treiber eine Kollision für die Einwegfunktion GDI finden.

In der folgenden Grafik ist der Vorgang dargestellt.



Die optimale Lösung um minimale Datengrössen zu erreichen wäre sicher, die Daten mehrfach durchzugehen, sie umzuordnen und neu zu speichern. Ein Desingziel war es aber, dass der DPS Treiber ein vollwertiger Windows Druckertreiber sein sollte, der sich an alle Regeln und Gebote hält, die ein guter Treiber einhalten sollte, um sich optimal ins System einzugliedern und maximale Systemstabilität zu gewährleisten. Dazu gehört auch, sich an die Grundgedanken eines Treibermodells zu halten. Für Windows legt dieses nahe, einen Druckertreiber so zu bauen, dass er jeden eingehenden Befehl direkt verarbeitet und weiterleitet. Daher kam eine mehrstufige Lösung nicht in Frage.

Der DPS Druckerporttreiber für Windows

Wie der Druckertreiber stellt auch der Druckerporttreiber (der Druckmonitor, wie er in Windows korrekt heisst), der für DPS geschrieben wurde, einen Spezialfall eines Ports dar. Genau wie der Drucker ist er nicht als Hardware vorhanden und seine Eigenschaften sind daher erst zur Lauf-

zeit bekannt. Er weiss erst wohin er die Daten senden muss, wenn er sie bereits vom Spooler empfängt.

Jeder Druckauftrag muss unter Umständen an einen anderen Server weitergeleitet werden. Daher ist es nicht möglich, den Port auf irgendeine Weise zu konfigurieren. Ansonsten würde er unter Umständen die falschen Aufträge an die falschen Server senden (die Sendephase kann durch die Pufferung des Spoolers sehr lange nach dem Rendering stattfinden).

Die Konfiguration wird vom Druckertreiber daher als erster Datenblock im Datenstrom übermittelt. Sobald der Port sie empfangen hat, konfiguriert er sich und baut alle benötigten Verbindungen auf.

Weil der Spooler ein spezieller Systemprozess ist und der Druckerport im Prozess des Spoolers ausgeführt wird, lassen sich Verbindungen nicht so einfach aufbauen, wie man es sich von der normalen Applikationsentwicklung her gewohnt ist[3]. Gewisse API-Aufrufe bringen das System zum Blockieren, ohne einen ersichtlichen Grund. Gewisse andere Aufrufe funktionieren dagegen problemlos. Leider ist es undokumentiert wieso das so ist.

Sehr wenige Probleme gibt es mit der lokalen Kommunikation. Deshalb startet der Port zuerst einen kleinen Proxyprozess und leitet sämtliche Kommunikation über diesen weiter.

Eine weitere Aufgabe des Druckerports besteht ausserdem darin, den Datenstrom je nach Konfiguration zu komprimieren. Verwendet wird hierzu eine Variante der LZx Algorithmen, welche mit einem einzigen Durchgang arbeitet. Sie ist schnell, gehört jedoch nicht zu den Spitzenreitern in Sachen Kompressionsrate.

[3] Das Konfigurationsmenü des Druckertreibers (der einzige Teil des Druckertreibers, der Daten sendet und empfängt), wird hingegen im Prozess der aufrufenden Anwendung ausgeführt. Sie hat daher nicht mit Einschränkungen zu kämpfen.

DPS Wiedergabe im Detail

Das Druckmodul

Das Gegenstück zur Aufzeichnung bildet das Druckmodul, das auf dem Server installiert ist. Es erhält den Datenstrom, der zuvor durch den Druckertreiber erstellt worden ist und interpretiert diesen. Es lädt temporär alle eingebetteten Schriftarten und erzeugt künstlich dieselbe Umgebung, die auch auf dem Clientcomputer gefunden werden kann. Dann druckt es den Auftrag, genau so, als wäre er auf dem Clientcomputer ausgeführt worden.

Neben der 1:1 Wiedergabe von Druckjobs beherrscht das Modul auch andere Funktionen. Die zuvor vom Client gewünschten Aufgaben wie z.B. das Drucken mehrerer Seiten pro Blatt oder das Spiegeln der Ausgabe, werden ebenfalls vom Druckmodul übernommen. Es ist auch für die Anordnung der Ausgabe auf dem Papier verantwortlich oder stellt die Druckreihenfolge wie gewünscht zusammen.

Die Schwierigkeit bei diesen Aufgaben liegt hauptsächlich darin, dass die temporär geladenen Umgebungseigenschaften, wie z.B. die Schriftarten, erst dann aus dem System entfernt werden dürfen, wenn der lokale Spooler die Daten durch den lokalen Druckertreiber geschickt hat und sie in die Druckersprache umgewandelt worden sind. Daher muss das Modul mit mehreren Phasen arbeiten, die anhand der Druckerüberwachung, welche der Warteschlangendienst betreibt, ausgelöst werden.

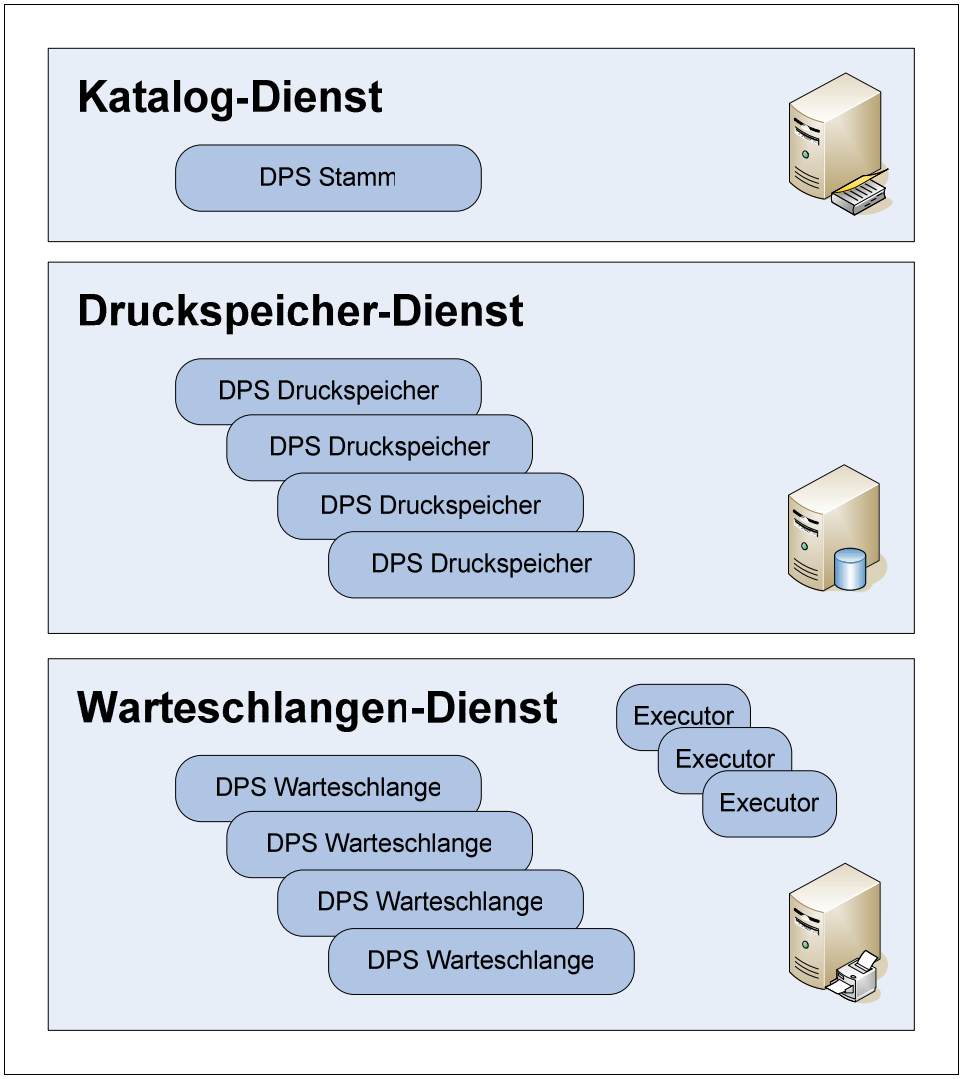
Die DPS Serverdienste

Dienste hinter der Struktur

Zu DPS gehören drei verschiedene Server-Dienste. Es gibt den Katalog-Dienst (Catalogue-Service), den Druckspeicher-Dienst (Storage-Service) und den Warteschlangen-Dienst (Queue-Service). Jeder dieser Dienste betreibt einen bestimmten Typ von Serverobjekt, welcher jeweils eine Repräsentation eines logischen DPS-Elements darstellt.

Genau wie die logischen Objekte besitzt auch jeder Dienst eine eindeutige ID in Form einer GUID. Somit kann jede Instanz eines logischen Objektes genau identifiziert werden. Die ID einer Instanz setzt sich aus 2 GUIDs zusammen (<GUID-Dienst><GUID-logisches Objekt>). Es ist wichtig, dass eine Instanz eindeutig erkannt werden kann (siehe *„Navigieren und Identifizieren“* und *„Jobs identifizieren“*).

Grafische Übersicht



Der Katalog-Dienst

Jeder Stamm der DPS Struktur hat genau eine Repräsentation auf genau einem Server und pro Server ist maximal ein Stamm zugelassen. Dieser Server fungiert als Katalog, um andere Objekte, welche zur Nutzung von DPS nötig sind, zu finden. Er ist eine Art DNS-Server, der die Clients mit

den IDs und den Adressen der Server versorgt, die eine Instanz eines bestimmten Objektes verwalten.

Der Katalogdienst ist der zentrale Dienst in DPS. Jede Serverinstanz eines anderen Objekts muss sich zuerst beim Stamm anmelden und muss daher zuerst eine Verbindung zum Katalog-Dienst herstellen. Die Adresse des Katalog-Dienstes muss daher fix sein und sie ist bei den einzelnen Serverobjekten registriert.

Die Clients verbinden sich mit dem Druckdienst, indem sie zuerst eine Verbindung zum Katalog-Dienst herstellen. Dann rufen sie über den darauf instanzierten Stamm alle benötigten Daten ab, um mit den Druckspeichern und Warteschlangen zu kommunizieren.

Meistens hat der Computer, auf dem der Katalog-Dienst läuft einen speziellen Namen wie z.B. „dps.ethz.ch“.

In der aktuellen Version besteht keine Möglichkeit den Stamm über mehrere Katalog-Dienste zu verteilen, wie das mit den Druckspeichern und Warteschlangen möglich ist. Sollte es nötig sein, in sehr grossen Umgebungen dies zu gewährleisten, könnte dies aber nachträglich eingebaut werden. Favorisiert würde dann eine Integration des DPS Stammes in ActiveDirectory, um die Synchronisation der einzelnen Computer durchzuführen.

Der Druckspeicher-Dienst

Der Druckspeicher-Dienst (Storage Service) ist eines der Kernstücke des ganzen Systems. Sämtliche Daten, welche für die Druckjobverarbeitung vom Client zu einem Server übertragen werden müssen, werden über den Druckspeicher-Dienst übermittelt.

Wie bereits erwähnt dient der Druckspeicher-Dienst dazu, Serverinstanzen von Druckspeicherobjekten zu beherbergen. Ein Computer, auf dem der Dienst läuft kann dabei im Gegensatz zum Katalog-Dienst mehr als nur eine Instanz verwalten. Da der Druckspeicher-Dienst nicht zu einem bestimmten Stamm gehört, können diese Instanzen auch zu verschiedenen Stämmen gehören. Durch die Verwendung global eindeutiger Namen für diese Objekte, wird nie eine Verwechslung der Objekte eintreten. Zudem garantiert das umfassende Sicherheitskonzept der Serverobjekte, dass niemand an Daten oder Druckaufträge gelangt, die ihm nicht zustehen.

Die getrennte Architektur des Speicherdienstes vom ausführenden Dienst wurde ganz bewusst gewählt. Neben den bereits erwähnten Vorteilen im Falle einer Störung in der Übertragung von Daten, bietet diese

Architektur mehr Möglichkeiten für verschiedene, an die Situation angepasste Implementationen. Falls in schnellen Netzwerken kleinere Druckaufträge verarbeitet werden sollen, bieten sie keine grossen Vorteile gegenüber einem Dienst, der beide Aufgaben vereint. Wird DPS aber für grosse Aufträge, über das Internet oder eine andere langsame Verbindung genutzt, dann bietet diese Architektur einen Vorteil: Durch die klare Trennung könnte mit Front-End Storage-Servern gearbeitet werden, welche die Aufträge von den Clients empfangen und erst die kompletten Daten über ein schnelles Netzwerk an die eigentlichen Server weiterleiten. So können die eigentlichen Daten-Server besser ausgenutzt werden, was am Ende zu geringeren Betriebskosten führt.

Im Grunde ist der Druckspeicher-Dienst eine auf grosse Binärdatenblöcke ausgerichtete Datenbank. Bevor ein Druckauftrag zur Ausführung vermerkt werden kann, muss er als Druckspeicher-Objekt auf einem Druckspeicher-Server abgelegt werden

Im Detail arbeitet der Druckspeicher-Dienst wie eine Datenbank. Er führt eine Logdatei um zu gewährleisten, dass ACID eingehalten wird. Nach einer bestimmten Zeit führt ein Hintergrundprozess immer wieder eine Sicherung der Daten durch. Käme es zu einem Absturz, gingen keine Daten verloren, welche zuvor einem Client als ‚empfangen' bestätigt worden sind (ausser es gäbe wirklich ein Problem mit der Hardware des Servers).

In der aktuellen Version ist allerdings die Funktion deaktiviert, welche ACID auch beim Empfang der Druckdaten garantieren könnte. Das System beschränkt sich darauf ACID für die Erstellung, das Löschen und das Modifizieren von Druckjobs und deren Eigenschaften zu garantieren. Wird das System auf einem NTFS Datenträger installiert, kann aber nach einem Absturz mindestens eine allfällige Beschädigung der Daten erkannt werden. Aus Performancegründen ist momentan nicht vorgesehen, diese zusätzliche Funktion zu aktivieren, da Druckaufträge selten so eminent wichtig sind, dass eine Ausführung garantiert werden muss.

Eine weitere Aufgabe des Druckspeicher-Dienstes besteht darin, im Hintergrund immer wieder dafür zu sorgen, dass im System keine Aufträge hängen bleiben. Der Betrieb wird daher dauernd überwacht und notfalls werden die Daten korrigiert. Dies übernimmt ein spezieller Garbage-Collection Prozess (siehe *„die Garbage-Collection"*).

Der Warteschlangen-Dienst

Der Warteschlangen-Dienst (Queue Service) ist die ausführende Einheit von DPS. Nachdem ein Druckauftrag heraufgeladen worden ist, reiht der

Client das Objekt in die Warteschlange eines Servers ein, der vom Warteschlangendienst verwaltet wird.

Sobald alle Bedingungen erfüllt sind, welche für die Ausführung eines Auftrags nötig sind, wird er an einen Executor übergeben. Ein Executor ist das ausführende Element. Es ist kein Teil des Serverobjekts, sondern global auf der Maschine vorhanden. Wenn mehrere Serverobjekte auf einer Maschine vorhanden sind, kann somit die gesamte Last einen gewissen Wert trotzdem nicht überschreiten.

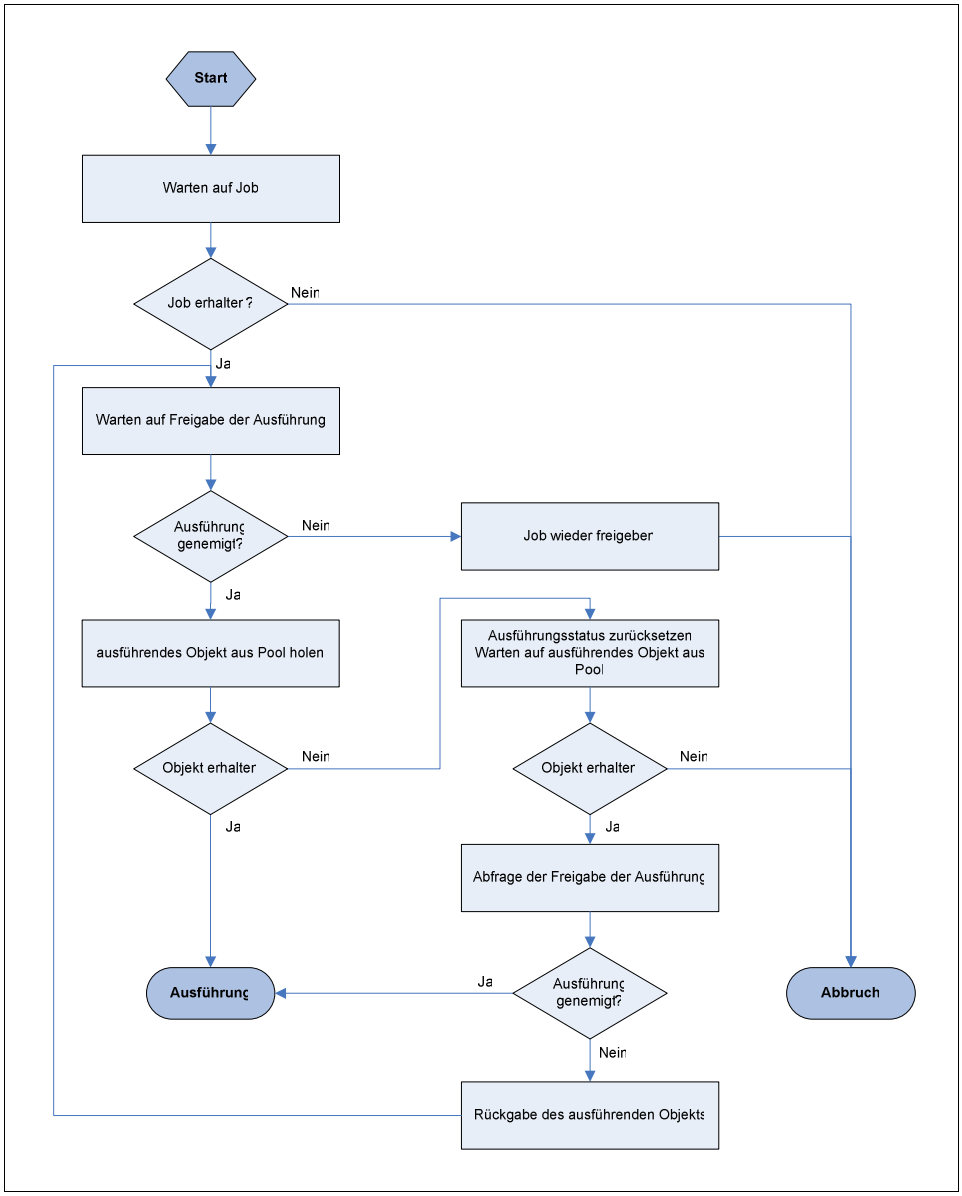
Die Warteschlange arbeitet dabei wie in der Grafik auf der nächsten Seite beschrieben ist. Die Ausführungsgenehmigung wird von einem Objekt erteilt, das den Zieldrucker überwacht. Erst wenn er weniger als eine bestimmte Zahl Druckaufträge in seiner Warteschlange hat, wird die Ausführung genehmigt.

Sobald ein Auftrag in einer Warteschlange platziert wurde, übernimmt diese alle Verwaltungsaufgaben für die Objekte, die zum Druckauftrag gehören. Sie löscht zum Beispiel auch die Datendatei, nachdem der Auftrag ausgeführt wurde und informiert Clients über den aktuellen Status des Auftrages.

Dabei arbeitet auch die Warteschlange wie eine Datenbank. Auch sie garantiert ACID und arbeitet mit Logdateien für ein allfälliges Recovery nach einem Absturz. Da Drucker eigene Speicher besitzen ist das schwieriger, da es passieren könnte, dass ein Auftrag komplett übertragen wurde und Windows meldet, der Auftrag sei gedruckt. Daraufhin wird die Warteschlange das so vermerken. Wenn jetzt ein Stromausfall eintritt, kann der Druck unterbrochen werden, obwohl er bereits als gedruckt vermerkt wurde. Dies lässt sich leider nicht umgehen. Nur wenn der Druckertreiber des realen Druckers das echte Ende des Druckauftrages anzeigt, kann DPS richtig arbeiten, ansonsten bleibt hier immer ein kleines Restrisiko, dass der Auftrag verloren gehen kann. Neuere Treiber bieten allerdings die nötige Unterstützung, um das echte Druckende anzuzeigen.

Um das Problem abzuschwächen kann DPS angewiesen werden, die Druckaufträge verzögert zu löschen, damit dieses Problem manuell gelöst werden kann und ein Druckauftrag neu eingereiht werden könnte.

Es ist auch eine Aufgabe des Warteschlangen-Dienstes, im Hintergrund immer wieder dafür zu sorgen, dass im System keine Aufträge hängen bleiben. Der Betrieb wird daher dauernd überwacht und notfalls werden die Daten korrigiert. Dies übernimmt auch hier ein spezieller Garbage-Collection Prozess (siehe *„die Garbage-Collection“*).


Start
Warten auf Job
Job erhalter ?
Nein
Ja
Warten auf Freigabe der Ausführung
Ausführung genemigt?
Nein
Job wieder freigeber
Ja
ausführendes Objekt aus Pool holen
Ausführungsstatus zurücksetzen
Warten auf ausführendes Objekt aus Pool
Objekt erhalter
Nein
Objekt erhalter
Nein
Ja
Ja
Abfrage der Freigabe der Ausführung
Ausführung
Ja
Ausführung genemigt?
Abbruch
Nein
Rückgabe des ausführenden Objekts

Implementationsdetails

Gegenseitiger Ausschluss

Bei Anwendungen mit mehreren Threads, besteht eine Schwierigkeit darin, den gegenseitigen Ausschluss beim Zugriff auf Systemressourcen, Dateien, Programmobjekte oder Variabeln zu garantieren. In den meisten Fällen wird das so gelöst, dass eine Anwendung vor dem Zugriff auf ein bestimmtes Objekt dieses sperren und am Ende wieder freigeben muss. Wird versucht ein Objekt zu sperren, das bereits gesperrt ist, dann wird der Thread so lange unterbrochen, bis das Objekt wieder freigegeben wurde.

Durch diese Methode kann es vorkommen, dass ein Objekt zum Flaschenhals wird und die ganze Leistung, welche durch die Parallelisierung gewonnen werden konnte, wieder aufgefressen wird. Oft kommt es bei Standardlösungen vor, dass Objekte auch dann gesperrt werden, wenn zwei Threads dasselbe Objekt nur lesen wollen.

Die Dienste von DPS arbeiten viel mit zentralen Arrays für Auftrags- oder Objektlisten und die meisten Zugriffe sind nur Lesezugriffe. Selten wird jedoch auch geschrieben. Um die Datenbank möglichst effizient zu gestalten, wurde deshalb ein spezieller Ausschlussmechanismus implementiert, der dem Umstand Rechnung trägt, dass lesende Threads gleichzeitig arbeiten können.

Will ein Thread ein Objekt auslesen, wird ihm der Zugriff gewährt, sofern niemand auf dieses Objekt schreibt, oder darauf wartet, schreiben zu dürfen.

Wenn ein Thread schreiben will, dann wird er blockiert, wenn bereits ein anderer Thread am schreiben ist, oder wenn ein Thread am Lesen ist. Ab diesem Zeitpunkt werden alle weiteren Lese-Anfragen blockiert. Egal ob dem Thread das Schreibrecht gewährt wurde oder nicht. Erst wenn er schreiben konnte, können andere wieder lesen (oder schreiben).

Die Blockierung im Falle eines für Schreibzugriffe gesperrten Objektes funktioniert dabei mit den normalen Mechanismen, die man von Standard-Ausschlussverfahren her kennt. Muss ein Thread, der gerade das Schreibrecht anfordern will, wegen aktiven Lesezugriffen gesperrt werden, dann wird eine Art „Busy-Waiting“ verwendet. Kurz nach der Aktivierung des Threads durch den Scheduler, prüft er ob der Zugriff jetzt gestattet ist und gibt andernfalls die Ausführungsberechtigung wieder ab. Der Aufwand sollte dadurch minimal sein, zumal Schreibzugriffe nicht sehr häufig auftreten.

Als weitere Optimierung wurde versucht, die Schreibzugriffe so kurz wie möglich zu halten und wenn möglich so wenige Objekte wie möglich zu sperren. Schreibzugriffe wie das Hinzufügen eines Objektes zu einem Array dauern dabei meist nicht lange. Muss aber ein Array gesperrt werden um ein Objekt zu löschen, dann arbeitet DPS in zwei Phasen. In der ersten Phase wird der Array zum Lesen gesperrt um die Position des Elements zu finden. Ist es gefunden, wird die Lesesperre aufgehoben und eine Schreibsperre beantragt. Wenn diese gewährt wurde, dann wird die zuvor gefundene Position überprüft und gegebenenfalls korrigiert. Dadurch bleibt die Liste nur kurz durch einen Schreibzugriff gesperrt, wodurch fast immer gelesen werden kann und selten Threads blockiert werden.

Die Garbage-Collection

Ein System, das mit Clients kommuniziert, muss immer damit rechnen, dass einer dieser Clients einen Fehler macht. Wenn z.B. ein Client einen Druckauftrag als Objekt auf dem Druckspeicher anlegt, ihn aber dann nie in einer Warteschlange platziert, dann wird der Auftrag ewig hängen bleiben. Ein Druckauftrag muss daher eine Art Timeout haben. Wird er nicht innerhalb einer nützlichen Frist (z.B. einer Stunde) platziert, so wird er gelöscht. Um diese Timeouts zu überwachen, besitzt jeder der drei DPS Dienste einen Garbage-Collection Prozess.

Der Prozess hat auf jedem Dienst eine andere, aber ähnliche Aufgabe. Der Katalog-Dienst führt bei seinen Aufräumaktionen eine Prüfung aller registrierten Objekte durch. Beim Start eines Serverobjekts, muss sich dieses am Katalog anmelden und mit der aktuellen Adresse registrieren. Beim Herunterfahren muss es sich wieder abmelden. Wenn nun aber die Verbindung unterbrochen wird, kann es sein dass der Dienst dies nicht kann. Damit der Katalog nun nicht dauernd falsche Daten an die Clients verteilt, werden diese von Zeit zu Zeit überprüft und notfalls korrigiert.

Im Druckspeicherdienst hat der Garbage-Collection Prozess die Aufgabe, verwaiste Druckaufträge zu löschen, die über eine längere Zeit nicht verwendet worden sind. Es wird dabei gewährleistet, dass nur die Aufträge gelöscht werden, die in keiner Warteschlange eingereiht sind. Wenn ein Auftrag allerdings in einer Warteschlange angemeldet wurde, diese aber die Anmeldung nicht bestätigen kann, wird der Auftrag ebenfalls entfernt.

Am wenigsten zu tun hat der Garbage-Collection Prozess beim Warteschlangen-Dienst. Hier gibt es praktisch keine Objekte, die ohne konkre-

ten Nutzen hängen bleiben können. Allerdings gibt es auch hier gecachte Daten, welche geprüft und notfalls korrigiert werden.

Der Hintergrundprozess

Jeder Server besitzt eine Garbage-Collection und muss seine Sicherheitsobjekte auf dem aktuellen Stand halten. Zudem gibt es noch einige andere Hintergrundarbeiten, die ab und zu durchgeführt werden müssen.

Für all diese Prozesse könnte jeweils ein eigener Thread gestartet werden. Da diese Aufgaben aber nicht zeitkritisch sind und im Falle einer gleichzeitigen Ausführung das System nur unnötig belasten würden, wurde eine andere Ausführungsart angewendet. Alle Aufgaben werden durch Events ausgelöst und ein einzelner Thread arbeiten diese dann hintereinander ab. Da diese Events nicht sehr häufig auftreten ist das bezüglich der Leistung oder den allfälligen Verzögerungen bei der Ausführung überhaupt kein Problem.

Sollte es unerwartet doch zu einer grösseren Verzögerung der Ausführung kommen, wird so lange kein neuer Event mehr ausgelöst, bis die bereits ausgelösten erledigt worden sind.

Das Sicherheitssystem

Jedes Objekt in Windows hat eine eigene Sicherheitsbeschreibung. Diese regelt, wem der Zugriff auf das Objekt gewährt und wem der Zugriff verweigert wird. Dabei können feine Abstufungen vorgenommen werden und z.B. Lese- sowie Schreibrechte einzeln vergeben werden.

Ein Beispiel für ein solches gesichertes Objekt ist eine Datei. Windows führt für jede Datei ein eigenes Sicherheitsobjekt. Genau so macht es auch DPS: Jeder Dienst, jeder Server und jedes Objekt hat eine solche Sicherheitsbeschreibung.

Wenn ein Client beim Stamm nach vorhandenen Druckern fragt, wird zuerst geprüft, ob er überhaupt das Recht hat auf den Stamm zuzugreifen. Danach prüft das System, welche Drucker er verwenden darf und sendet dann eine Liste dieser Drucker. Einen Drucker den er nicht verwenden darf, wird der Benutzer daher nicht zu sehen bekommen.

Wenn der Client eine Verbindung zum Druckspeicher aufbaut, wird auch hier geprüft, ob er berechtigt ist Daten in diesen Speicher zu schreiben; erst dann wird ein Objekt erzeugt. Dieses Objekt wird jetzt mit den speziellen Zugangsdaten des erstellenden Benutzers gesichert. Standardmässig darf nur der Benutzer oder ein Administrator dieses Objekt verwenden (also drucken), anzeigen oder löschen. Auch den Status kann niemand anders abrufen.

Dieses Prozedere wiederholt sich noch einmal wenn der Client den Druckauftrag in der Warteschlange platziert. Auch das Auftragsobjekt in der Warteschlange wird auf diese Weise gesichert.

Während die Sicherheitsobjekte für die vom Client erstellten Objekte bei der Erstellung generiert werden, werden Sicherheitseinstellungen, welche die Serverobjekte betreffen, zentral vom Katalog verwaltet. Ein Administrator kann auf dem Katalog festlegen, welches Objekt wem welchen Zugriff gewährt oder verweigert.

Meldet sich ein Server am Katalog an, ruft er als erstes diese Sicherheitseinstellungen ab und wendet sie an. Während der Laufzeit werden diese Informationen zusätzlich von einem Hintergrundprozess (ähnlich dem Garbage-Collection Prozess) überprüft und wenn nötig angepasst bzw. neu vom Katalog heruntergeladen.

Durch dieses strenge Sicherheitskonzept ist ein Missbrauch ausgeschlossen. Es kann auf kein Element zugegriffen werden, ohne dass die Sicherheit überprüft würde.

Das Postscript-Relais

In Unix gibt es nur den Postscript-Weg um zu drucken. Alle Druckaufträge sind in ihrer ursprünglichen Form in diesem Format vorhanden. Es gibt heute für Unix zwar Software, die es ermöglicht, auch nicht postscriptfähige Drucker an das System anzuschliessen, Diese arbeitet aber mit einem Filter, der das Postscript in andere Daten umwandelt, bevor sie an den Drucker gesendet werden. Eines dieser Systeme ist CUPS (Common Unix Printing System).

Seit Apple die Entwicklung völlig eigenständiger Betriebssysteme aufgegeben hat und heute nur noch Lösungen, die auf Unix basieren, anbieten, kann CUPS auch auf Apple Computern betrieben werden. Weil die beiden grössten nicht Microsoft-Betriebssysteme mit CUPS umgehen können, bot es sich an, einen Zusatz zu CUPS zu entwickeln, um die Druckaufträge (im Postscriptformat) zu einem Windows System umzuleiten.

Um die Aufträge auf diesem Windows System verarbeiten zu können, müssen sie von Postscript in das DPS-Format VPD umgewandelt werden. Diese Aufgabe übernimmt das Postscript-to-DPS Relais.

Weil CUPS, im Gegensatz zu Windows, nur statische Konfigurationsmenüs zulässt, boten sich hier leider nicht dieselben Möglichkeiten zur Konfiguration der Druckausgabe, wie in Windows. Wenn aber die richtigen Eigenschaftsdateien zum gewünschten Ausgabedrucker generiert worden sind, dann funktioniert das System recht benutzerfreundlich.

Das Postscript-to-DPS Relais kann auch ohne den CUPS-Teil verwendet werden. Es ist relativ einfach anzusteuern und könnte z.B. auch als eine Übergangslösung dienen, um einen Postscript-Druckdienst auf DPS umzustellen.

Zusammenfassung

Bei den Distributed Printing Services (DPS) handelt es sich um ein komplettes Drucksystem, das dafür gedacht ist, Aufträge von verschiedenen Computern aufzuzeichnen und auf einem Server wieder abzuspielen.

Der Benutzer sieht nur noch einen virtuellen Drucker, mit dem er den ganzen Dienst bedienen kann. Er muss sich nicht mehr um die tatsächlichen Drucker kümmern. Zudem sind alle Informationen über den Status eines Auftrags online abrufbar.

Das System entspricht ausserdem den Anforderungen an die Sicherheit, wie sie heute notwendig sind. Es gibt keine ungesicherten Zugriffe auf Dienste, Objekte oder Daten.

DPS lässt sich von einem einzigen Arbeitsplatz aus konfigurieren und überwachen. Der Betrieb kann so effizient gestaltet werden und Ausfälle werden sofort erkannt.

Weitere Arbeiten

DPS wurde einer kleinen Gruppe von Testpersonen zugänglich gemacht. Als nächstes wird das System ausgiebig getestet und möglicherweise vorhandene Fehler beseitigt. Anschliessend könnte das Produkt auch grossflächiger in den Einsatz kommen.

Das Produkt befindet sich jetzt im Zustand eines „Release Candidate“. Wenn es von den Testpersonen positiv bewertet wird, kann es zu einem ersten Release werden. Ansonsten müsste es nochmals überarbeitet werden.

Es ist zu hoffen, dass es in naher Zukunft eingesetzt werden kann und dass es bald schon auf mehr Computern, als nur den bisherigen Testmaschinen anzutreffen ist.