List of Abbreviations

API	Application Program Interface
AVI	Audio Video Interleave
CPU	Central Processing Unit
DLL	Dynamic Link Library
IDE	Integrated Drive Electronics
IP	Internet Protocol
IPC	InterProcess Communication
ISDN	Integrated Services Data Network
JDK	Java Development Kits
JNI	Java Native Interface
JVM	Java Virtual Machine
LAN	Local Area Network
MAN	Metropolitan Area Network
OS	Operating System
OSI	Open System Interconnection
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SCSI	Small Computer System Interface
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
USB	Universal Serial Bus
VfW	Video for Windows
WAN	Wide Area Network
WLAN	Wireless Local Area Network

Abstract

With the recent growth and development of the network and internet usage in colleges, companies and other network places, it is important to find some way to improve the reliability and efficiency of using these networks. Because of the large-scale growth in the number of users and peripheral devices used in these places such as (printers, scanners, and cameras), there are a variety of problems which have to be solved, such as remote peripheral device is stopped for a specific reason.

To solve these problems and guarantee the good performance and continuous maintenance of peripheral devices in the network, a monitoring system can be used for this purpose.

This work concerned with the implementation of a monitoring system on peripheral devices called Device Status Monitoring System (DSMS). It is a network monitoring type that could be classified as fault and accounting monitoring system.

The aim of this research is to monitor one or more remote peripheral devices which are connected to the network PC(s) in a LAN. The proposed (DSMS) can monitor the status of printer, scanner, and camera. In addition, it saves the printer information on special files which can be used later by the Administrator.

To implement DSMS, two programming language were used JAVA and VC++. Java language is used to support network activities, while VC++ deals with operating system API. Java Native Interface (JNI) is used to combine these two languages

Acknowledgment

First of all I would like to express my sincere gratitude and appreciation to my supervisors Dr. Lamia H. Khalid and Dr. Sawsan Kamal for their valuable guidance, supervision and untiring efforts during the course of this work.

Special thanks to the College of Science, dean of the college for the continuous supports and encouragement during the period of my studies.

Grateful thanks for the Head of Department of Computer Science Dr. Taha S. Bashaga, staff and employees.

Finally, my very special thanks to my family, my husband, and my friends especially, M.Sc. group for their continuous supports and encouragement during the period of my studies.

Reem

Appendix A

A.1 JOB_INFO_1

The **JOB_INFO_1** structure specifies print-job information such as the job-identifier value, the name of the printer for which the job is spooled, the name of the machine that created the print job, the name of the user that owns the print job, and so on.

```
typedef struct _JOB_INFO_1 {
  DWORD  JobId;
  LPTSTR pPrinterName;
  LPTSTR pMachineName;
  LPTSTR pUserName;
  LPTSTR pDocument;
  LPTSTR pDatatype;
  LPTSTR pStatus;
  DWORD  Status;
  DWORD  Priority;
  DWORD  Position;
  DWORD  TotalPages;
  DWORD  PagesPrinted;
  SYSTEMTIME Submitted;
} JOB_INFO_1, *PJOB_INFO_1;
```

Members

JobId

Specifies a job identifier.

pPrinterName

Pointer to a null-terminated string that specifies the name of the printer for which the job is spooled.

pMachineName

Pointer to a null-terminated string that specifies the name of the machine that created the print job.

pUserName

Pointer to a null-terminated string that specifies the name of the user that owns the print job.

pDocument

Pointer to a null-terminated string that specifies the name of the print job (for example, "MS-WORD: Review.doc").

pDatatype

Pointer to a null-terminated string that specifies the type of data used to record the print job.

pStatus

Pointer to a null-terminated string that specifies the status of the print job. This member should be checked prior to **Status** and, if **pStatus** is NULL, the status is defined by the contents of the Status member.

Status

Specifies the job status. This member can be one or more of the following values.

Value	**Meaning**
JOB_STATUS_BLOCKED_DEVQ	The driver cannot print the job.
JOB_STATUS_DELETED	Job has been deleted.
JOB_STATUS_DELETING	Job is being deleted.
JOB_STATUS_ERROR	An error is associated with the job.
JOB_STATUS_OFFLINE	Printer is offline.
JOB_STATUS_PAPEROUT	Printer is out of paper.
JOB_STATUS_PAUSED	Job is paused.
JOB_STATUS_PRINTED	Job has printed.
JOB_STATUS_PRINTING	Job is printing.
JOB_STATUS_RESTART	Job has been restarted.
JOB_STATUS_SPOOLING	Job is spooling.
JOB_STATUS_USER_INTERVENTION	Printer has an error that requires the user to do something.

Priority

Specifies the job priority. This member can be one of the following values or in the range between 1 through 99 (MIN_PRIORITY through MAX_PRIORITY).

Value	**Meaning**
MIN_PRIORITY	Minimum priority.
MAX_PRIORITY	Maximum priority.
DEF_PRIORITY	Default priority.

Position

Specifies the job's position in the print queue.

TotalPages

Specifies how many pages the document contains. This value may be zero if the print job does not contain page delimiting information.

PagesPrinted

Specifies the number of pages that have printed. This value may be zero if the print job does not contain page delimiting information.

Submitted

A **SYSTEMTIME** structure that specifies the time that this document was spooled. This time value is in Universal Time Coordinate (UTC) format.

A.2 PRINTER_INFO_2

The **PRINTER_INFO_2** structure specifies detailed printer information.

```
typedef struct _PRINTER_INFO_2 {
  LPTSTR    pServerName;
  LPTSTR    pPrinterName;
  LPTSTR    pShareName;
  LPTSTR    pPortName;
  LPTSTR    pDriverName;
  LPTSTR    pComment;
  LPTSTR    pLocation;
  LPDEVMODE pDevMode;
  LPTSTR    pSepFile;
  LPTSTR    pPrintProcessor;
  LPTSTR    pDatatype;
  LPTSTR    pParameters;
  PSECURITY_DESCRIPTOR pSecurityDescriptor;
  DWORD     Attributes;
  DWORD     Priority;
  DWORD     DefaultPriority;
  DWORD     StartTime;
  DWORD     UntilTime;
  DWORD     Status;
  DWORD     cJobs;
  DWORD     AveragePPM;
} PRINTER_INFO_2, *PPRINTER_INFO_2;
```

Members

pServerName

Pointer to a null-terminated string identifying the server that controls the printer. If this string is NULL, the printer is controlled locally.

pPrinterName

Pointer to a null-terminated string that specifies the name of the printer.

pShareName

Pointer to a null-terminated string that identifies the sharepoint for the printer. (This string is used only if the PRINTER_ATTRIBUTE_SHARED constant was set for the **Attributes** member.)

pPortName

Pointer to a null-terminated string that identifies the port(s) used to transmit data to the printer. If a printer is connected to more than one port, the names of each port must be separated by commas (for example, "LPT1:,LPT2:,LPT3:").

Windows 95: This member can specify only one port because multiple ports per printer are not supported.

pDriverName

Pointer to a null-terminated string that specifies the name of the printer driver.

pComment

Pointer to a null-terminated string that provides a brief description of the printer.

pLocation

Pointer to a null-terminated string that specifies the physical location of the printer (for example, "Bldg. 38, Room 1164").

pDevMode

Pointer to a **DEVMODE** structure that defines default printer data such as the paper orientation and the resolution.

pSepFile

Pointer to a null-terminated string that specifies the name of the file used to create the separator page. This page is used to separate print jobs sent to the printer.

pPrintProcessor

Pointer to a null-terminated string that specifies the name of the print processor used by the printer. You can use the **EnumPrintProcessors** function to obtain a list of print processors installed on a server.

pDatatype

Pointer to a null-terminated string that specifies the data type used to record the print job. You can use the **EnumPrintProcessorDatatypes** function to obtain a list of data types supported by a specific print processor.

pParameters

Pointer to a null-terminated string that specifies the default print-processor parameters.

pSecurityDescriptor

Pointer to a **SECURITY_DESCRIPTOR** structure for the printer. This member may be NULL.

Windows 95: This member is ignored.

Attributes

Specifies the printer attributes. This member can be any reasonable combination of the following values.

Value	**Meaning**
PRINTER_ATTRIBUTE_DEFAULT	**Windows 95:** Indicates the printer is the default printer in the system.
PRINTER_ATTRIBUTE_DIRECT	Job is sent directly to the printer (it is not spooled).
PRINTER_ATTRIBUTE_DO_COMPLETE_FIRST	If set and printer is set for print-while-spooling, any jobs that have completed spooling are scheduled to print before jobs that have not completed spooling.
PRINTER_ATTRIBUTE_ENABLE_BIDI	**Windows 95:** Indicates whether bi-directional communications are enabled for the printer.
PRINTER_ATTRIBUTE_ENABLE_DEVQ	If set, **DevQueryPrint** is called. **DevQueryPrint** may fail if the document and printer setups do not match. Setting this flag causes mismatched documents to be held in the queue.

PRINTER_ATTRIBUTE_KEEPPRINTEDJOBS	If set, jobs are kept after they are printed. If unset, jobs are deleted.
PRINTER_ATTRIBUTE_QUEUED	If set, the printer spools and starts printing after the last page is spooled. If not set and PRINTER_ATTRIBUTE_DIRECT is not set, the printer spools and prints while spooling.
PRINTER_ATTRIBUTE_SHARED	Printer is shared.
PRINTER_ATTRIBUTE_WORK_OFFLINE	**Windows 95:** Indicates whether the printer is currently connected. If the printer is not currently connected, print jobs will continue to spool.
PRINTER_ATTRIBUTE_PUBLISHED	**Windows 2000:** Indicates whether the printer is published in the directory service.
PRINTER_ATTRIBUTE_NETWORK	Printer is a network printer connection.
PRINTER_ATTRIBUTE_HIDDEN	Reserved.
PRINTER_ATTRIBUTE_LOCAL	Printer is a local printer.
PRINTER_ATTRIBUTE_RAW_ONLY	Indicates that only raw data type print jobs can be spooled.

Priority

Specifies a priority value that the spooler uses to route print jobs.

DefaultPriority

Specifies the default priority value assigned to each print job.

StartTime

Specifies the earliest time at which the printer will print a job. This value is expressed as minutes elapsed since 12:00 AM GMT (Greenwich Mean Time).

UntilTime

Specifies the latest time at which the printer will print a job. This value is expressed as minutes elapsed since 12:00 AM GMT (Greenwich Mean Time).

Status

Specifies the printer status. This member can be any reasonable combination of the following values.

Value	**Meaning**
PRINTER_STATUS_BUSY	The printer is busy.
PRINTER_STATUS_DOOR_OPEN	The printer door is open.
PRINTER_STATUS_ERROR	The printer is in an error state.
PRINTER_STATUS_INITIALIZING	The printer is initializing.
PRINTER_STATUS_IO_ACTIVE	The printer is in an active input/output state
PRINTER_STATUS_MANUAL_FEED	The printer is in a manual feed state.

PRINTER_STATUS_NO_TONER	The printer is out of toner.
PRINTER_STATUS_NOT_AVAILABLE	The printer is not available for printing.
PRINTER_STATUS_OFFLINE	The printer is offline.
PRINTER_STATUS_OUT_OF_MEMORY	The printer has run out of memory.
PRINTER_STATUS_OUTPUT_BIN_FULL	The printer's output bin is full.
PRINTER_STATUS_PAGE_PUNT	The printer cannot print the current page. **Windows 95:** Indicates the page is being "punted" (that is, not printed) because it is too complex for the printer to print.
PRINTER_STATUS_PAPER_JAM	Paper is jammed in the printer
PRINTER_STATUS_PAPER_OUT	The printer is out of paper.
PRINTER_STATUS_PAPER_PROBLEM	The printer has a paper problem.
PRINTER_STATUS_PAUSED	The printer is paused.
PRINTER_STATUS_PENDING_DELETION	The printer is deleting a print job.
PRINTER_STATUS_POWER_SAVE	The printer is in power save mode.
PRINTER_STATUS_PRINTING	The printer is printing.
PRINTER_STATUS_PROCESSING	The printer is processing a print job.
PRINTER_STATUS_SERVER_UNKNOWN	The printer status is unknown.
PRINTER_STATUS_TONER_LOW	The printer is low on toner.
PRINTER_STATUS_USER_INTERVENTION	The printer has an error that requires the user to do something.
PRINTER_STATUS_WAITING	The printer is waiting.
PRINTER_STATUS_WARMING_UP	The printer is warming up.

cJobs

Specifies the number of print jobs that have been queued for the printer.

AveragePPM

Specifies the average number of pages per minute that have been printed on the printer.

Chapter One

Overview

Chapter Two

Computer Network and Monitoring Concepts

Chapter Three

Implementation of Device Status Monitoring System (DSMS)

Chapter Four
DSMS Interface and Testing

Chapter Five

Conclusion and Future Work

Appendix A

References

Dedication

To My Beloved Family

Reem

بسم الله الرحمن الرحيم

(يَرْفَعْ اللَّهُ الَّذِينَ آمَنُوا مِنْكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ وَاللَّهُ بِمَا تَعْمَلُونَ خَبِيرٌ)

صدق الله العظيم

المجادلة(١١)

Reference

[Ans06] Answers Corporation web site, "Video for Windows", 2006. http://www.answer.com/topic/video for windows information from Answer_com.htm

[App02] Apple Computer, Inc., "Introduction to the Mac OS X Printing System", 2002. http://developer.apple.com/documentation/printing/conceptual/ bout_MacOSX_Printing/index.html

[Bec04] British educational communication and technology agency, "network monitoring project", Becta 2004. www.becta.org.uk/tsas/docs/fits-netmon.pdf

[Bra03] Bradley L., "Low Paper Monitoring System ", 2003. http://seniord.ee.iastate.edu/may2003/dr.doc

[Bur04] Burgess M., "Principles of Network and System Administration", John Wiley and Sons Ltd, Second Edition, 2004.

[Col04] Colin Goldsmith, "wireless local area networking for device monitoring", M.Sc. thesis, Department of Electrical and Computer Engineering, University of Rochester New York, 2004. www.ece.rochester.edu/research/wcng/papers/thesis/goldsmith-MSthesis.pdf

[Cou01] Coulouris G., Dollimore J., Kindberg T., "Distributed System", Addison-Wesly, Third edition, 2001.

[Cra99] Crage Z. and Paul D., Bookshelf, "Upgrading and Repairing Networks Internet", Macmillan Computer Publishing, 1999.

[Dan99] Dan W. Blacharski and John Enck, "Managing Multivendor Networks", second edition, 1999.

[Dei01] Deitel H. M. and Deitel P. J., "Java How to Program", Fourth edition, published by Prentice Hall, 2001.

[Dou99] Douglas E. Comer, "Computer Networks and Internet", Second Eddition, 1999.

[Edm00] Edmund W., "network monitoring fundomentals and standards report", 2000. http://www.cis.ohio-state.edu/~jain/cis788-97/net-monitoring/index.htm

[Fai00] Fairhurst G., "The OSI Reference Model", 2000.
http://www.erg.abdn.ac.uk/users/gorry/course/osi.html

[Fre04] FreePatentOnline web site, 2004.
http://www.freepatentsonline.com/5317693.htm

[Fuj01] FUJITSU, "Scanner Utility for Microsoft Windows Version 9.6 software", 2001.
http://tech.nikoyo.com.cn/manual/drvguide9.pdf

[Jav05] The Java^{TM} Tutorial, Sun Microsystems, Inc., 2005.

[Led01] Leduc G., "network monitoring", 2001.
www.montefiore.ulg.ac.b/~leduc/cource/ISIR/ISIR-chap2.pdf

[Lit00] Litwak K., "Pure Java 2", Sams Publishing, 2000.

[Lui99] Luiz A. DaSilva, "Network Management paper", Center for Wireless Telecommunication, Virginia Polytechnic Institute and State University, 1999.
www.ee.vt.edu/~prangapr/documants/rangaprabhu-Qos_paper.doc

[Man05] ManageEngine^{TM}, "OpManager 5.5 product", AdventNet, Inc., 2005.
http://manageengine.adventnet.com/products/OpManager

[Mat98] Matt H., "SAMS teach yourself networking in 24 hours", Sams publishing first edition, 1998.

[Mho00] MSDN Microsoft corporation, "How To: Get the Status of a Printer and a Print Job", 2000.

[Mim00] MSDN Microsoft corporation, "Imaging Supports", 2000.

[Mvi00] MSDN Microsoft corporation, "Video capture device driver channels", 2000.

[Opt06] Optio software company, Inc., "Optio Print Manager™", 2006.
http://www.optiospftware.com/default.aspx?tabid=66

[Pac04] Patrick P., "LPRng SourceForge Project ", AStArt Technologies, advanced state of the art, computer and network technologies, 2004.
http://www.lprng.com/#lprngdist

[Par99] Parker T., Bookshelf, "Teach Yourself TCP/IP in 14 Days", Sams Publishing, Second Addition, 1999.

[Pra97] Prashant S., "Advance java networking", published by prentice-Hall, Inc., 1997.

[Pre05] Premio website, Inc., 2005.
www.premiopc.com

[Ser05] ServerFiles website, "software directory for network administrator and IT proc", 2005.
http://www.serverFiles.com/network monitoring software.htm

[Tan03] Tanenbaum S. Andrew, "Computer Network", Pentice-Hall, inc., Fourth Edition, 2003.

[Tim05] Timo L., "Print Support for MHP", M.Sc. Thesis, Helsinki university of technology, department of Automation and system technology, 2005.
www.media.hut.fi/~julkaistut/diplomityot/DI_T_Laitinen_2005.pdf

[Twa06] TWAIN working group web site, 2006.

[Wik06] Wikipeia site, "the free Encyclopedia", Inc., 2006.
http://en.wikipedia.org/wiki/Peripheral_device

[Wkp06] Wikipeia foundation site (free Encyclopedia), Inc., "Video for Windows", 2006.
http://en.wikipedia.org/wiki/video_for_windows.htm

[Woo97] Woodhull S. Albert and Tanenbaum S. Andrew, "operating systems design and implementation", second edition, 1997.

Table of Contents

جمهورية العراق
وزارة التعليم العالي و البحث العلمي
جامعة النهرين
كلية العلوم

تصميم و تنفيذ نظام مراقبة و توثيق العمل الخاطيء للاجهزة

رسالة مقدمة الى كلية العلوم، جامعة النهرين كجزء من متطلبات نيل شهادة الماجستير في علوم الحاسوب

من قبل
ريم باقر جعفر الصفار

بكالوريوس
٢٠٠٣

المشرفون

د. لمـــياء حافظ خالد
د. سوسن كمال ثامر

تشرين الاول ٢٠٠٦
رمضان ١٤٢٧

Republic of Iraq
Ministry of Higher Education and scientific research
Al-Nahrain University
College of Science

Fault and Accounting Components Monitor System Design and Implementation

A Thesis
Submitted to the College of Science, Al-Nahrain University
In Partial Fulfillment of the Requirements for
The Degree of Master of Science in Computer Science

By
Reem Baqer Jaffar Al-Saffar
(B.Sc. 2003)

Supervisors
Dr. Lamia H. Khalid　　　　Dr. Sawsan K. Thamer

October 2006　　　　Ramadhan 1427

الخلاصة

مع النمو المتزايد و التطور في استخدام الشبكات و الانترنت في الكليات، الشركات و غيرها، اصبح من الضروري ايجاد طريقة ما لتحسين اداء و كفاءة استخدام هذه الشبكات. بسبب النمو المتزايد في عدد المستخدمين و الاجهزة الطرفية المستخدمة في هذه الاماكن مثل الطابعات، السكنرات، و الكاميرات، هناك مشاكل متنوعة يجب حلها، مثلاً توقف جهاز طرفي بعيد عن العمل لسبب معين.

لحل هذه المشاكل و ضمان الاداء الجيد، و الصيانة المستمرة للاجهزة الطرفية في الشبكة، من الممكن استخدام نظام مراقبة لهذا الغرض.

هذا العمل يهتم بتنفيذ نظام مراقبة للاجهزة الطرفية يسمى نظام مراقبة حالة الاجهزة (Device Status Monitoring System) (DSMS). هذا النظام يصنف ضمن انظمة مراقبة الخطأ و تسجيل التقارير (fault and account monitoring).

الهدف من هذا البحث هو مراقبة جهاز طرفي بعيد او اكثر و الذي يكون مربوطاَ بأجهزة الحاسوب على الشبكة المحلية (LAN).النظام المقترح (DSMS) ممكن ان يراقب حالة الطابعة ، السكانر، و الكاميرا. بالاضافة الى ذلك، يقوم بخزن معلومات عن الطابعة في قواعد بيانات خاصة، و التي من الممكن ان يستخدمها المراقب في وقت لاحق.

من اجل تنفيذ (DSMS)، تم استخدام لغتي برمجة و هما JAVA و VC++. لغة JAVA تستعمل من اجل دعم فعاليات الشبكة، بينما VBC++ تتعامل مع API الخاصة بنظام التشغيل. Java Native Interface (JNI) استخدمت للربط بين هاتين اللغتين.

Chapter Four

DSMS Interface and Testing

4.1 Installing DSMS

DSMS interface designed to enable the Administrator to monitor the peripheral devices in the host or remote PC(s) easily. DSMS system consists of two parts, Administrator part which is installed in the Administrator's computer, and the Agent part which is installed in the remote computers.

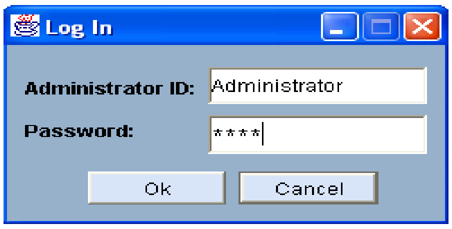
The Agent.exe must run in the remote PC in the network by adding it to the start up menu of the remote PC which has no interface with the user. At the Administrator side, the Administrator.exe file is executed in the Administrators computer to start monitoring.

4.2 Running DSMS

DSMS system is designed with six frames to enable the Administrator to monitor the local or remote peripheral devices, these frames are: Log In frame, DSMS frame, Lists frame, Computers List frame, Devices List frame, and Printers Archive frame.

4.2.1 Password frame

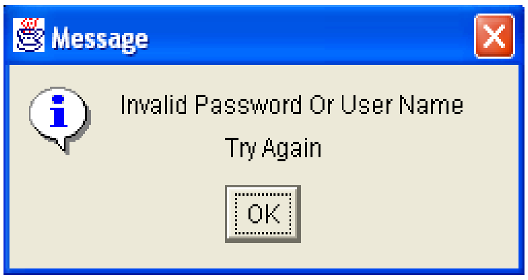
When DSMS executed, the first frame, as shown in figure (4.1), will appear to check the authenticity of the Administrator.



Administrator ID: Administrator
Password: ****

Figure (4.1): Log In frame.

When the Administrator enters the correct ID and password, the next frame will appear on the Administrator screen, but if the Administrator enters wrong ID or wrong password or both, warning message is appear as shown in figure (4.2). The system gives the user three trails to enter the correct ID and password. If the user fails, the executed system ends.



Invalid Password Or User Name
Try Again

Figure (4.2): Warning message.

4.2.2 DSMS frame

This frame is arising on the Administrator screen when the Administrator enters the correct Administrator ID and password. It contains a menu of two options **Monitoring** and **Help**, these options illustrated below:

1. **Monitoring** option: This option contains two items, **Start** and **Exit** as shown in figure (4.3).
    - **Start**: To search for switched-ON PC(s) on the LAN and to search which device(s) are connected with each PC, click the **Start** item in the **Monitoring** option as shown in figure (4.3).
    - **Exit**: this button to end the execution of the monitoring system.

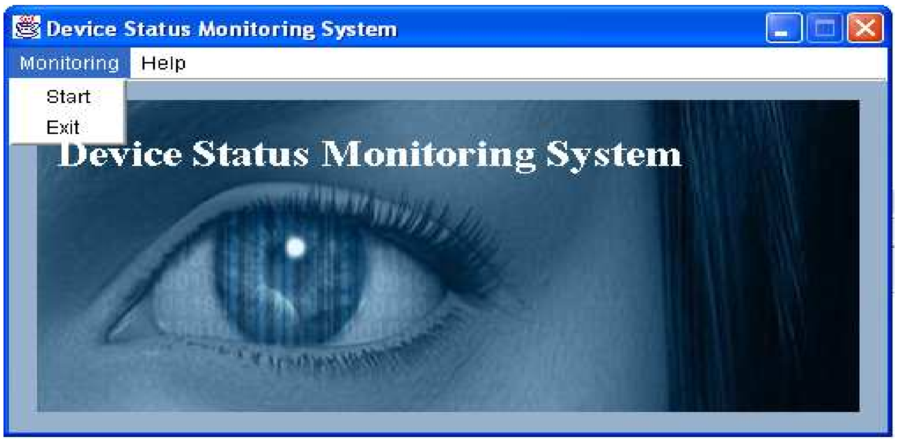

Device Status Monitoring System
Monitoring
Help
Start
Exit
Device Status Monitoring System

Figure (4.3): Monitoring option.

2. **Help**: This option contains one function, this function is **About**, as shown in figure (4.4).
    - **About**: This function displays a message box contain short information about the DSMS, as shown in figure (4.5).

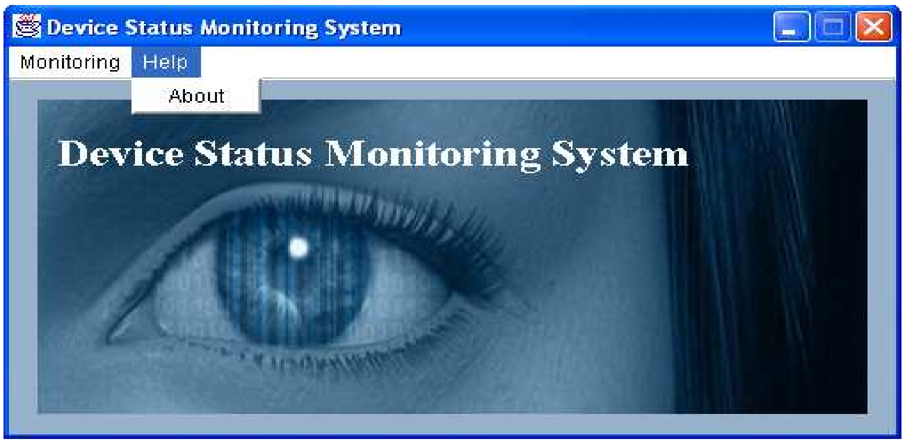
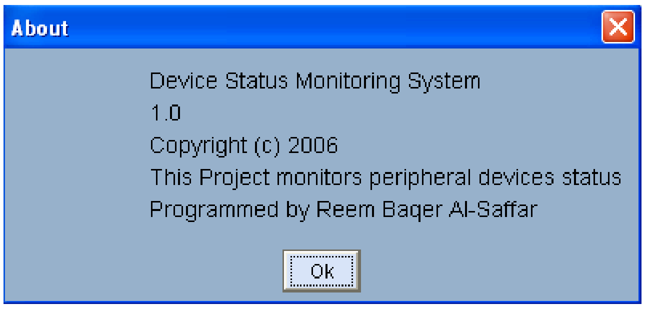

Device Status Monitoring System
Monitoring
Help
About
Device Status Monitoring System

Figure (4.4): Help option.



Device Status Monitoring System
1.0
Copyright (c) 2006
This Project monitors peripheral devices status
Programmed by Reem Baqer Al-Saffar

Figure (4.5): About frame.

4.2.3 Lists frame

After completing the search for the computers and devices, the new frame, as shown in figure (4.6), is created. This frame consists of two buttons: **Computers List** button and **Devices List** button. These buttons display the information of monitoring, shown in section (4.2.4) and section (4.2.5).

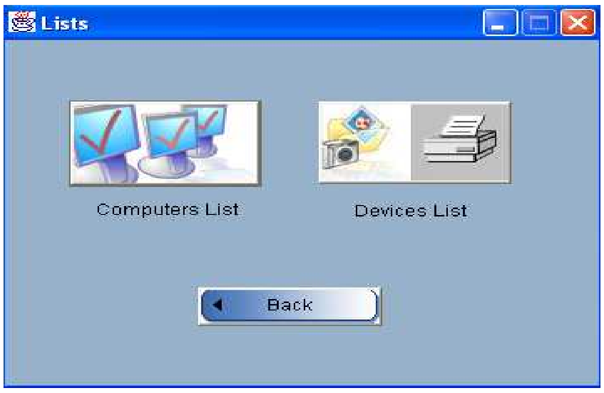
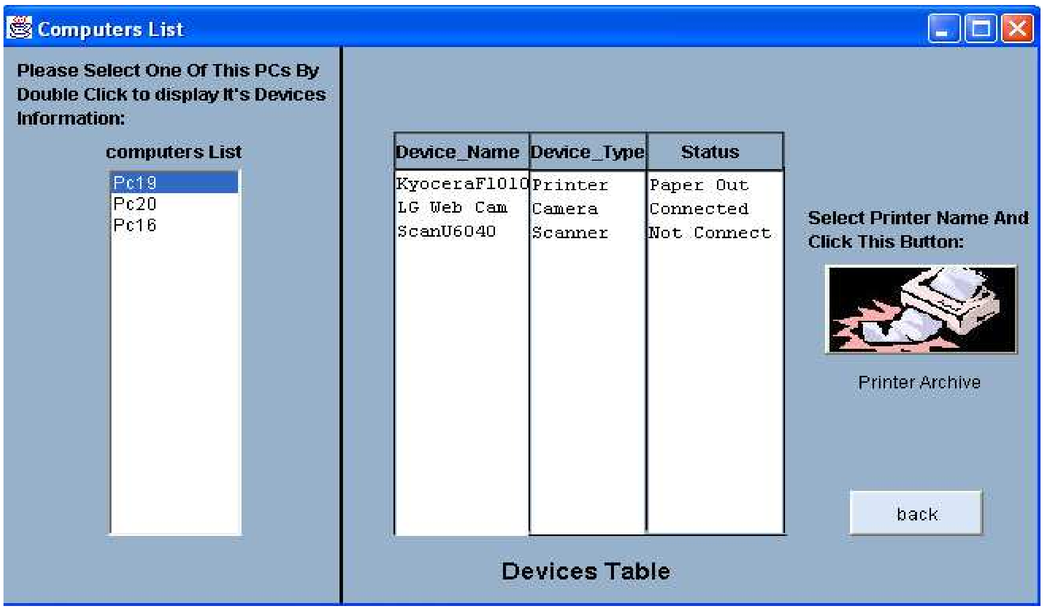


Figure (4.6): Lists Frame.

4.2.4 Computers List frame

When the **Computers List** button in Lists frame is chosen, a new frame, as shown in figure (4.7), will appear to display the information about the devices of each computer.


Computers List
Please Select One Of This PCs By Double Click to display It's Devices Information:
computers List
Pc19
Pc20
Pc16
Select Printer Name And Click This Button:
Printer Archive
back
Devices Table

Device_Name	Device_Type	Status
KyoceraF1010	Printer	Paper Out
LG Web Cam	Camera	Connected
ScanU6040	Scanner	Not Connect

Figure (4.7): Computers list frame.

This frame contains: Computers List, Devices table, Printer Archive button, and back button.

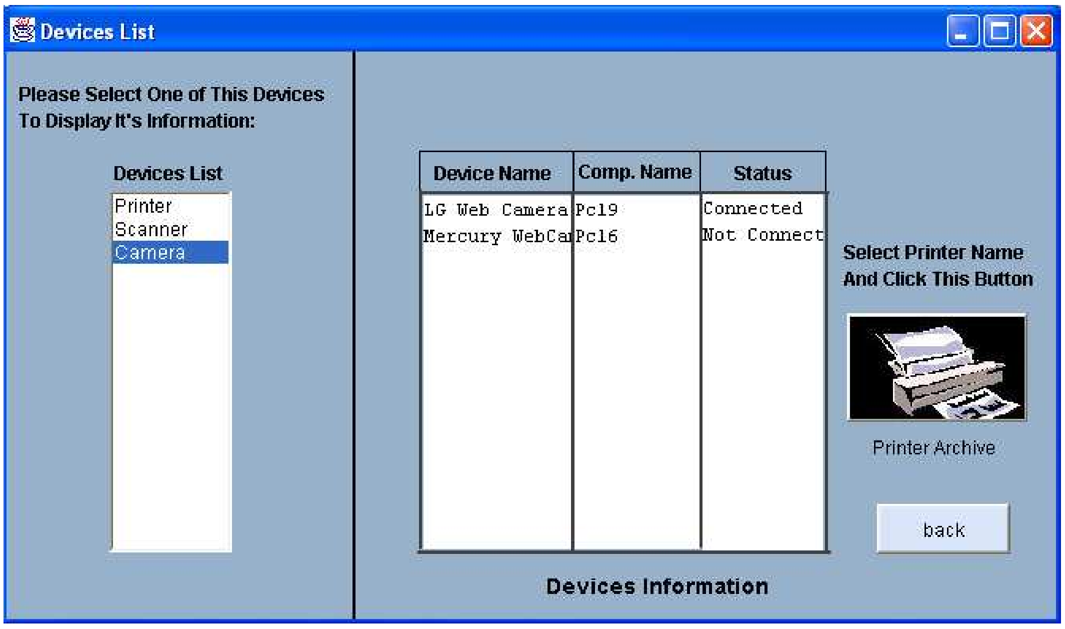
- Computers List: displaying the switched-ON computers on the network.
- Devices table: this table contains the information of the devices of the computers. It consists of three columns, Device_Name, Device_Type, and Status.

  When any PC name is chosen from the Computers List, a table is displayed containing:

  a. Devices name (driver name or manufacture name).
  b. Devices type (this device either printer or scanner or camera).
  c. And the status of each device.
- Printer Archive button: displays the information of a specific printer, as shown in section (4.2.6).
- Back button: this button back the user to the Lists frame.

4.2.5 Devices List frame

When the **Devices List** button in the List frame is chosen, Device List frame, as shown in figure (4.8), will appear, to display the information about the peripheral devices in the LAN.



Device Name	Comp. Name	Status
LG Web Camera	Pc19	Connected
Mercury WebCa	Pc16	Not Connect

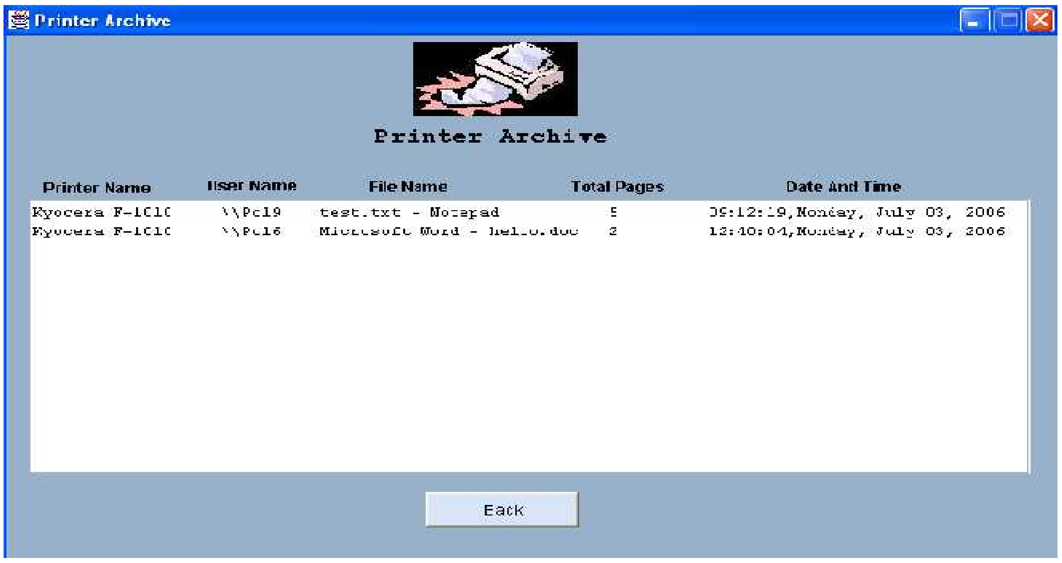
Figure (4.8): Devices list frame.

This frame contains: Devices List, Devices information table, Printer Archive button, and Back button.

- Devices List: display a list of monitored peripheral devices.
- Devices information table: this table contains the information about the selected device. It consists of three columns, Device_Name, Comp_Name, and Status.
  When a device is chosen from the Devices List, a table contains devices' names, their status, and the computers' names that own these devices.
- Printer Archive button: display the archive of a specific printer**,** as shown in section (4.2.6).
- Back button: this button back the user to the Lists frame.

4.2.6 Printer Archive frame

This frame displays the archive of the printer. To display this archive, the printer name is selected from the Device_Name column in the Computers List frame or Devices List frame and the **Printer Archive** button is clicked on, then the archive frame, as shown in figure (4.9), will be displayed on the screen.



Printer Name	User Name	File Name	Total Pages	Date And Time
Kyocera F-1010	\\Pc19	test.txt - Notepad	5	09:12:19,Monday, July 03, 2006
Kyocera F-1010	\\Pc16	Microsoft Word - hello.doc	2	12:40:04,Monday, July 03, 2006

Figure (4.9): Printer Archive frame.

The Printer Archive frame displays the table that contains information about the selected printer, this information is:

- Name of the host or remote printer user who send a print job to the printer spooler.
- File name to be printed.
- Total pages of this file to be printed.
- Date and time of printing.

4.3 DSMS Example

To run the DSMS system, the user of this system (Administrator) must follow many steps. These are:

Step1: When DSMS system starts, the Administrator must enter the correct ID and password in the Log In frame, as shown in figure (4.1), to enter the system.

Step2: The next frame is DSMS frame, as shown in figure (4.3). Click on the **Start** menu item to search for the switch-ON computer and receive the monitoring information.

Step3: The List frame, as shown in figure (4.6) is the next frame after completing the search. If the Administrator click on **Computers List** button then go to the step (4), else if click on **Devices List** button then go to the step (5).

Step4: The Computer List frame, as shown in figure (4.7), displays the list of computers and the Administrator must select one of the computers to display the table of the peripheral devices of this PC, or select one printer and click on **Printer Archive** button to display the archive of selected printer in the printer archive frame, as shown in figure (4.9).

Step5: The Computer List frame, as shown in figure (4.8), display the list of monitored peripheral devices and the Administrator must select one of this devices to display the table of the information of this device and which PC own this device, or select one printer and click on **Printer Archive** button to display the archive of selected printer in the printer archive frame, as shown in figure (4.9).

Chapter Five
Conclusion and Future Work

5.1 Conclusion

From this research, many things were noticed and concluded. The following are the most important ones:

1. In any client-server programming model, the multithreaded technique must be used in the server to ensure high response time in the client side.
2. VC++ language is a suitable language to implement the system operation because it have an instruction set that can get access to the Operating System to get information about the peripheral devices. While java language is suitable to implementing the network operation.
3. From IPC mechanisms point of view, the socket is the best technique to be used because:
    a. It is implemented with few statements and need few variables.
    b. It's powerful in sending and receiving traditional data types.
    c. It uses the TCP/IP protocol, so it can be used in internet programming.

During system implementation, several problems are raised, the main problems are:

- The first problem was finding the proper programming language that strongly supports all system and network functions in a perfect way. To solve this problem, more than one programming language (VC++ and Java) are used to take the benefits of each one. But a new problem arose which was how to combine these different programming languages. This problem has been solved by using Java Native Interface (JNI) method to combine java language with VC++.

- When monitoring the status of the peripheral devices, a problem appears that the types of a devices are different and the drivers of the devices also differs from device to another, when one driver read a status, another driver not read it. For example, the driver of the printer different from one printer to another therefore, not all status was read in all drivers. Also the types of the scanner are different and not all scanners give the same result.
- In Windows operating system, no DLL is available which supports the scanner monitoring. Therefore a written DLL was installed with the Windows DLL.
- TWAIN API was used to monitor the scanner, and this API caused the system to be run slowly.
- The archive was implemented to the printer only because the printer is a shared device, therefore a number of useful information can be recorded such as printer name, user name, printed file name, total pages of printed file, and date and time of printing. But the scanner and the camera are dedicated devices and they are used by the host PC user only.

5.2 Future work

After developing DSMS, several ideas come to mind that may improve the overall performance. These ideas have been left as recommendation for future work. These recommendations are:

1. Possible development for this work is by adding new peripheral devices for monitoring for example modem and speaker. Also, the internal components can be monitored such as hard disk, floppy, and motherboard of the host computer.
2. Running the proposed system on a Wide Area Network (WAN).
3. A development to DSMS system can be made by adding the network devices such as network printer and IP camera, where they are connected directly to the network not to a PC.
4. Add an interface to the managed devices configuration to add new printer, new scanner and so on.
5. Provide addition authentication and security to the Agent such that only the Administrator may access to the functions of Agent and the files of all monitored devices.

Chapter One

Overview

1.1 Introduction

A ***computer network***, like internet, exists whenever two or more computers are linked together through some type of communications medium.

Computer network is a shared resource used to exchange information between users. A computer network is a distributed collection of computers, viewed by the user as on large computers system which allocates jobs without use intervention [Cra99].

Network management includes the deployment, integration and coordination of the hardware, software and human elements to monitor, test, poll, configure, analyze, evaluate and control the network and element resources to meet the real-time, operational performance, and quality of service requirements at a reasonable cost.

Network management system is a software that allows to set up a ***network-management console***--or simply the ***Network Manager***. The Network Manager is the master control panel. With the Network Manager, software runs as agents on each device that wants to manage. The agents communicate with the Network Manager to provide all the information that needed to manage a device. Some agents are proprietary in nature and some follow standards, but generally both serve the same purpose: to interact with the managed device to find out its status and then report back to the Network Manager. While agent software varies from vendor to vendor, most provide the following basic functionality [Lui99]**:**

- Status information about the managed device.
- Statistics about the managed device's network interface(s).
- An interface to the managed device's configuration.
- The ability to detect faults in the managed device and report on them to the Network Manager. This reporting mechanism is often referred to as a trap.
- The ability to provide authentication and security to the agent such that only a Network Manager with the correct security may access the above functions on the agent.

Network Management can be used in the following Areas [Bur04]:

1. Performance management.
    - Objective: quantify, measure, report, analyze and control the performance (e.g., utilization, throughput) of different network components.
    - Components include individual devices (e.g., links, routers, and hosts) and end-end paths through the network.
    - Protocol standards such as the Simple Network Management Protocol (SNMP).
2. Fault management.
    - Objective: log, detect, and respond to fault conditions in the network.
    - Responsible for the immediate handling of transient network failures while performance management takes the longer term view of providing acceptable levels of performance in the face of

varying traffic demands and (hopefully rare) network device failures.

- SNMP plays a central role here also

3. Configuration management.
    - Objective: track which devices are on the managed network and, the hardware and software configurations of these devices.
4. Accounting management.
    - Objective: specify, log, and control user and device access to network resources.
    - Usage quotas, usage-based charging, and the allocation of resource access privileges fall under accounting management.
5. Security management.
    - Objective: control access to network resources according to some well-defined policy.
    - use of firewalls to monitor and control external access points to one's network is a component of security management

Network Management's main features and benefits include [Lui99]:

- Performance monitoring for switched environment.
- Multilayer switch monitoring.
- Comprehensive and easy-to-use device configuration.
- Backup and easy distribution of device configurations.
- Easy configuration of network traffic management policies.
- Easy updating of device software.
- Easy mapping of hosts to switch ports.

- Fault diagnosis and management.
- Remote access via the Internet.
- Multiple user access.

1.2 Network monitoring

Network monitoring is the information collection function of network management. The purpose of network monitoring is the collecting of useful information from various parts of the network so that the network can be managed and controlled using the collected information.

Most of the network devices are located in remote locations. These devices do not usually have directly connected terminals so that network management application cannot monitor their statuses easily. Thus, network monitoring is developed to allow network management applications to check the states of their network devices. As more and more network devices are used to build bigger networks, network monitoring is expanded to monitoring networks as a whole.

As more people communicate using networks, networks have become bigger and more complex. The proliferation of the internet has increased the pace of network expansions. At this age of big and complex networks, network monitoring applications need to use effective ways of checking the status of their networks, and network components so that network management applications can fully control their network and provide economical, and high-quality networking services to the users [Edm00].

Network and application monitoring systems which monitor hardware, software and network traffic are a must to determine problems, isolate bottlenecks and develop fixes. There are generally three basic goals for network monitoring [Edm00]:

- Performance monitoring deals with measuring the performance of the network.
- Fault monitoring deals with measuring the problems in the network. The objective of fault monitoring is to identify faults as quickly as possible after they occur and to identify the cause of the fault so that remedial action may be taken.
- Account monitoring deals with how users use the network.

1.3 Network monitoring benefits

There are many benefits that can be achieved by implementing network monitoring software in the environment. Here are some common benefits to have network monitoring software [Bec04]:

- Help ensure that the network is available to users.
- Respond more quickly to hardware and software incidents and problems.
- Determine where the network is performing well or otherwise – for example, where there might be bottlenecks in bandwidth, preventing parts of the network from operating efficiently, or at all (this is particularly important for networks where network usage often, follows an unusual pattern such as the peaks in use of the network).
- Identify trends and determine how to optimize the network by changing network configurations, replacing network devices and so on – this will help establishing where it is likely to obtain the best return on future investment in the network hardware and software.

- Improve the visibility of network status – will help to see instantly what parts of the network are working or not at any given time (this will help improving response times when part of the network does fail).
- Get timely notifications when failures occur.

1.4 Literature survey

Various efforts in the field of device monitoring system were introduced; some of these efforts are summarized below:

1. **B. Leitz, M. McWhirter, Low Paper Monitoring System, 2003.**

At the present time, printers only have the capability to alert users when the printer is out of paper. This project's focus is to devise an inexpensive system that will inform users about how much paper the printer has compared with the number of pages the user wants to print. The system will initially be implemented on a DeskJet type printer; however, portability to other platforms, such as other printers, fax machines, and copiers will be an important consideration during all aspects of the design [Bra03].

2. **Colin Goldsmith, Wireless Local Area Networking For Device Monitoring, 2004.**

In this thesis, a wireless local area networking technique is developed, which is intended for use in a device monitoring system. Wireless local area networking (WLAN) protocols (802.11, 802.11a, 802.11b, 802.11g, Bluetooth, HomeRF, and Ultrawideband) are quickly becoming a standard solution for connecting many different types of devices together. In this thesis, a WLAN will be used for a device monitoring application. The endless possibilities of wireless technologies shall be used to monitor the variables involved in printers, fax machines, scanners, and other devices.

Finally, a prototype of the wireless local area network for device monitoring is designed with code developed using the Java programming language [Col04].

3. Patrick Powell, LPRng SourceForge Project (LPRng - An Enhanced Printer Spooler), 2004.

The LPRng software is an enhanced, extended, and portable implementation of the Berkeley LPR print spooler functionality. The implementation is completely new and provides support for the following features: dynamic redirection of print queues; automatic job holding; highly verbose diagnostics; multiple printers serving a single queue; greatly enhanced security checks; and a greatly improved permission and authorization mechanism. The source software compiles and runs on a wide variety of UNIX systems, and is compatible with other print spoolers and network printers that use the LPR.

LPRngTool is a Graphical User Interface for the monitoring and configuration of the LPRng printing system. This tool is [Pac04]:

- Printer Monitoring:
  - Monitoring facility for spool queues.
  - Detailed queue and device status.

4. ManageEngine^{TM}, OpManager 5.5, 2005.

ManageEngine OpManager 5.5 product is a network monitoring software that combined WAN, Server and Application monitoring. OpManager automates several network monitoring tasks and removes the complexity associated with network management. OpManager can auto-discover entire network, group devices into intuitive maps, monitor devices in real-time, alert instantaneously on failure and provide comprehensive reports and graphs. One of the features of this product is printer monitoring.

Using OpManager printer monitoring functionality, operators can receive instant alerts when faults such as Paper jam occur. OpManager detects low-toner / no-toner, low paper / No paper scenarios [Man05].

5. ServerFiles.com, Print Inspector, 2005

ServerFiles.com is server software and hardware directory for Network administrators & IT professionals, listing networking and server software for Windows 2003, Windows 2000, NT and Linux; and now also listing networking hardware solutions focused on server based computing.

It produces many products, one of these products is Print Inspector. Print Inspector is a print management and auditing solution for the user corporate network. This software lets the user manage the print jobs queued to any shared printer and provides easy access to the printer and print server settings. Print Inspector also features print auditing capabilities: it saves to the special database detailed statistics about all printed documents (including the document name, date and number of pages, job date and time, name of the user who created the job, name of the computer from which the job was sent to the printer and more) [Ser05].

6. Optio Software, Optio Print Manager™, 2006.

This software helps the user to print more efficiently and dependably**.** With Optio Print Manager, the user will have the confidence that the documents that start, sustain and complete the critical business processes are printed -- including each page of every job.

Optio Print Manager will help the user to address these common printing challenges**:** assured delivery, printer availability, printer definition management, printer load balancing, automated printer failover, Unnecessary IS support calls, printing resource utilization [Opt06].

1.5 Aim of Thesis

The aim of this project is to implement network-monitoring system on peripheral devices and get the status of these devices as a client-server model. One or more remote devices in the network can be monitored using this system. The proposed system can monitor the devices which are connected to the remote PCs and display their status; in addition it saves to a special files detailed statistics about all printed documents (printer archive).

1.6 Thesis layout

In this section, the contents of individual chapters of this thesis are briefly reviewed.

- **Chapter two:** covers the theoretical basis of networking, monitoring system, and peripheral devices.
- **Chapter three:** presents proposed system architecture, and the algorithms that used to implements this system.
- **Chapter four:** presents the user interface for the designate system.
- **Chapter five:** introduces conclusion on this work, with recommendation for future work.

Chapter Three

Implementation of Device Status Monitoring System (DSMS)

3.1 Introduction

This work concerned with the implementation of a monitoring system called Device Status Monitoring System (DSMS). It is a network monitoring type that could be classified as fault and accounting monitoring system. DSMS monitors and reports the devices activates that are connected to a host or remote PC which work on a LAN. The proposed system depends on the concept of client/server system.

DSMS consists of two parts, the Agent and Administrator monitoring parts. The Agent part which resides on the remote PC is responsible to get the data (information) from the peripheral devices which are connected to the remote user PC and send it to the Administrator monitoring part on the Administrator PC. The Administrator monitoring part which is resided on the Administrator PC, in its turn, is responsible of displaying the information to the Administrator or saves to the special files detailed statistics about all printed documents (printer archive) for later use.

A device status monitoring system must be implemented using languages that support the networking operations, in addition to the Operating System APIs (Application Program Interface).

To implement DSMS, JAVA is chosen as a network operation support language, while Visual C++ as the language that has instruction set which provides access to the Operating System APIs to get information about peripheral devices status (printer status, scanner status, and camera status).

DSMS is run on a LAN with bus architecture at which all computers are connected to the network HUB, and all computers are running under windows XP operating system.

3.2 The DSMS Architecture

To build DSMS, two computers and one device, at least, must be exist and connected through the network. One PC should work as monitoring administrator and the other PCs with their peripheral (devices) are monitored by the administrator. Each PC that has peripheral device(s) connected with it installed with an Agent subsystem. The peripheral devices are classified into two types: the first type can be shared by different PCs in the network such as the printer. The second type is dedicated devices. They are used by the host computer only because this type of devices takes the input from the user directly such as the scanner and the camera while the printer takes the input from the computer. Figure (3.1) shows the architecture of the DSMS.

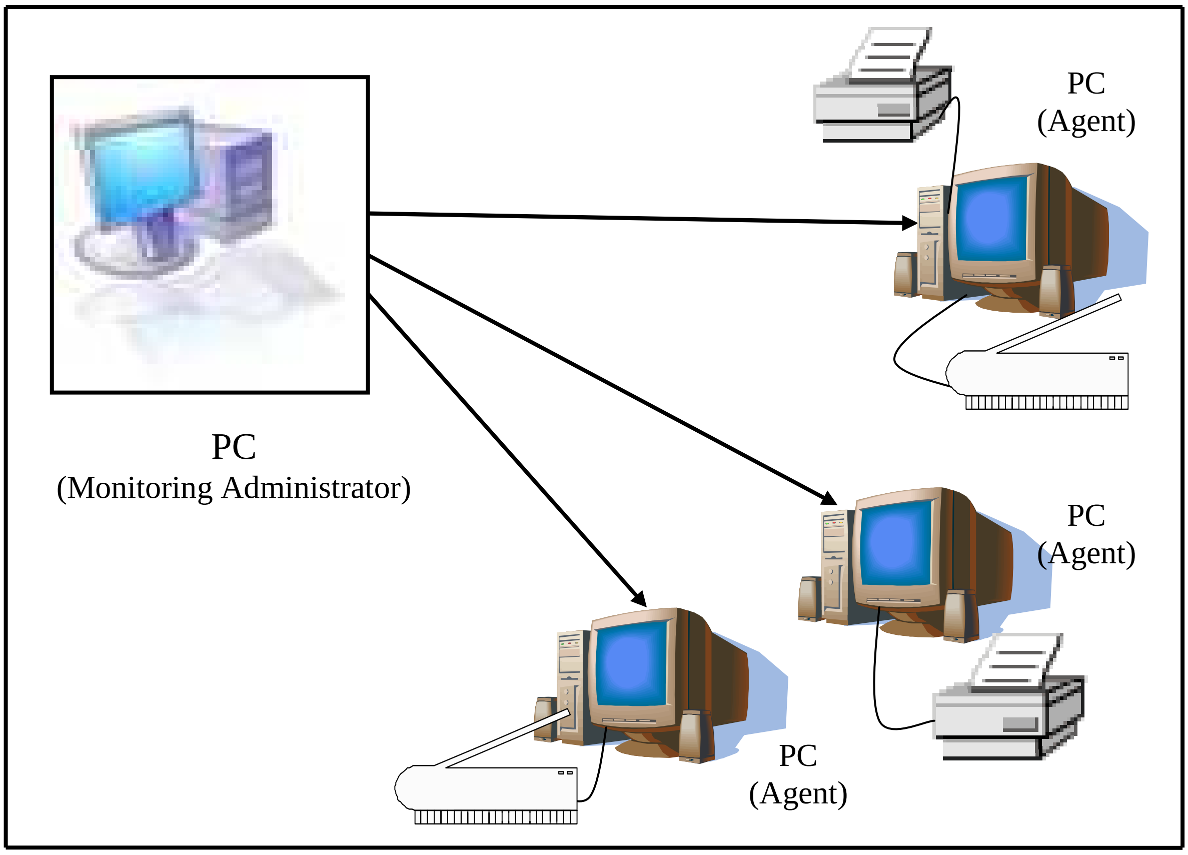


Figure (3.1): Devices Status Monitoring System (SDMS) Architecture

The Administrator and Agent subsystem of the DSMS perform many functions. The Agent subsystem performs the following functions:

- Printer monitoring.
- Scanner monitoring.
- Camera monitoring.
- Get printers archive.

The functions of the Administrator subsystem are:

- Identify the PCs which are connected with the LAN, and search for the devices which are connected with each PC.
- Start Monitoring.
- Select the PC to be monitored.
- Select the device to be monitored.

3.3 DSMS Model

The DSMS depends on client/server communication using sockets. The overview of client/server system is shown in figure (3.2).

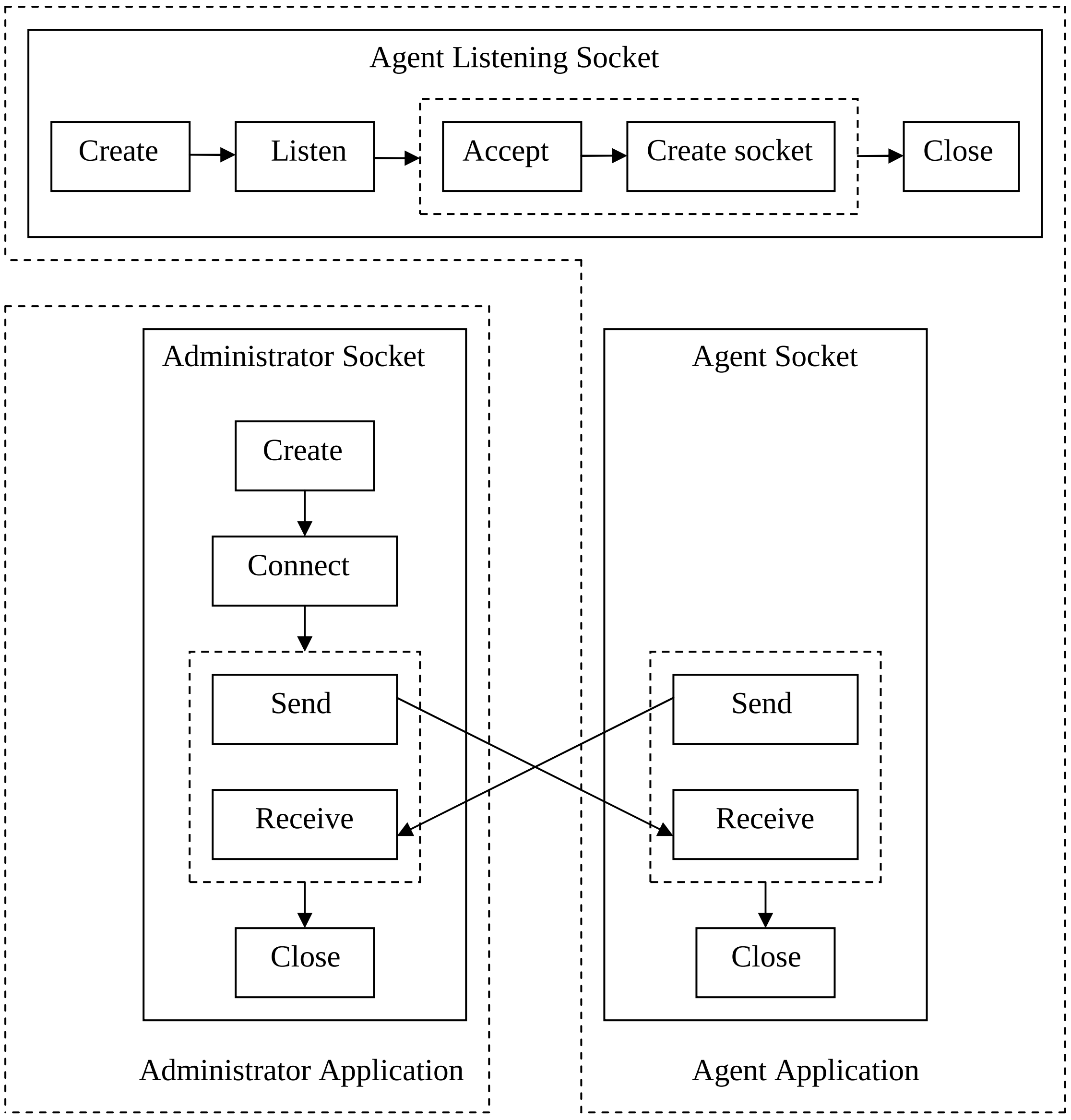


Figure (3.2): General client/server view.

The DSMS model consists of two parts: Administrator subsystem and Agent subsystem. Figure (3.3) shows DSMS model (Administrator and Agent).

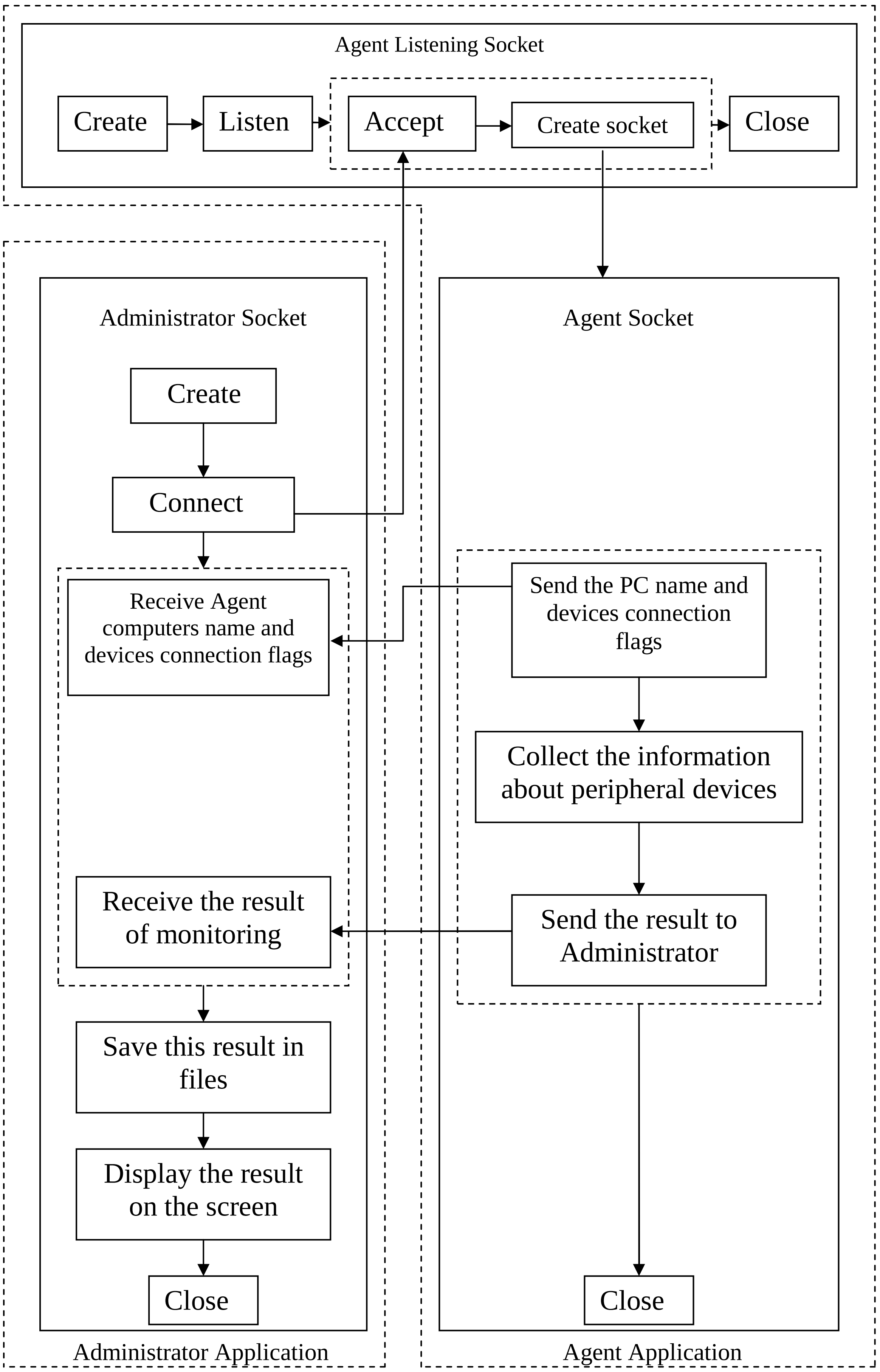

Agent Listening Socket
Create
Listen
Accept
Create socket
Close
Administrator Socket
Create
Connect
Receive Agent computers name and devices connection flags
Receive the result of monitoring
Save this result in files
Display the result on the screen
Close
Agent Socket
Send the PC name and devices connection flags
Collect the information about peripheral devices
Send the result to Administrator
Close
Administrator Application
Agent Application

Figure (3.3): DSMS model (Administrator and Agent subsystem).

3.3.1 The Agent subsystem

The Agent is the remote part of the monitoring system which is installed on the remote PC to be monitored. DSMS can create many Agents which are installed on each monitored PC. The Agent must run in the background silently without notification load or activity appearance so as not to disturb the user.

To prevent the user from interference with the Agent triggering, the Agent runs at windows startup in a standby mode waiting to receive the Administrator signal in order to start monitoring the device activates and sent the monitoring information to the Administrator. The Agent opens a socket and listening port, then waits a connection from the Administrator. The Administrator knows the ports number that the Agent listening to. When the connection established, the Agent starts its job by running appropriate monitoring functions (**printer monitoring thread, scanner monitoring thread, or camera monitoring thread**, **and get printer archive function**).

The main Agent module divides into two algorithms, the first algorithm open a connection with the Administrator to send the name of Agent PC to the Administrator and check which devices connected with PC. And the second algorithm waits a request from Administrator to monitor the devices. Algorithm (3.1) shows the check algorithm and algorithm (3.2) shows the main Agent algorithm.

Algorithm (3.1) check algorithm

Goal: This algorithm opens a connection with Administrator and check the connection of devices.

Input: None.

Output: None.

```
While (true)
{
  Open a connection with Administrator by open a socket(ID,Port_No).
```

Continued

```
    Send the PC name to the Administrator.
    Check the connection of printer and send a flag to the Administrator.
    Check the connection of scanner and send a flag to the Administrator.
    Check the connection of camera and send a flag to the Administrator.
}
```

Algorithm (3.2) main Agent algorithm

Goal: This algorithm opens a connection with Administrator and monitors the peripheral devices.

Input: None.

Output: None.

```
If  (printer is found)  then
    Set a Timer.
    Run Printer_Archive function (see algorithm (3.6)).
While (true)     //until shut down, restart, or stop the Agent
{
   Open server socket (port number).
   Wait a request of monitoring from Administrator.
   If  the flag of printer connection is true  Then
       Start Printer_Moitoring thread (see algorithm (3.3)).
       Send the result to Administrator.
   If  the flag of scanner connection is true  Then
       Start Scanner_Monitoring thread (see algorithm (3.4)).
       Send the result to Administrator.
   If  the flag of camera connection is true  Then
       Start Camera_Monitoring thread (see algorithm (3.5)).
       Send the result to Administrator.
}
```

The Agent consists of four modules. These modules are:

1. The printer monitoring module

The printer monitoring module is to get the printer's information by using the windows API, and these information are printer name and status (represent as a number). The printer status is one of these: (out of paper, not found {the printer not configured}, job of printer is paused, printer is switched OFF, the printer is printing, the job of printer is spooling, Job is deleting, no toner, and low toner), represent as list in the Administrator subsystem.

The Spooler of printer must be attempting to send a print job to the physical printer. This is the only time the state of the printer is reported by the port monitor. The most meaningful information about print jobs is reported by using a **JOB_INFO1** structure as shown in appendix (A). JOB_INFO structure describes a full set of values associated with a job and specifies print job information such as the job-identifier value, the name of the printer for which the job is spooled, the name of the machine that created the print job, the name of the user that owns the print job, and so on for that particular print job. **JOB_INFO1** structures are returned by **GetJob** API function. This function retrieves information about a print job. To determine the status of the printer by examining the status member of a **PRINTER_INFO2** structure as shown in appendix (A). **PRINTER_INFO2** structure specifies detailed printer information and the status value of a printer. This structure is returned by the **GetPrinter** function which retrieves information about a specified printer. The status member of the **PRINTER_INFO2** structure may be set with values that represent the status of the printer as a message such as **PRINTER_STATUS_PAUSED** or **PRINTER_STATUS_PAPEROUT** and so on.

Each of these functions requires a handle to a printer to identify the desired printer (printer handle is a number assigned to a printer that is used by the operating system to keep track of the attributes of that printer). This handle is obtained from **OpenPrinter** function; **OpenPrinter** function accepts a string containing the name of the printer as an input and returns the printer handle as an output. The printer name can be either local name of the printer or share name to a network printer.

Algorithm (3.3) shows the steps of the printer monitoring function which collects information about the printer.

Algorithm (3.3) printer monitoring algorithm

Goal: This algorithm monitors the printer and returns the printer information.

Input: None.

Output: Printers information file.

```
Step1:  Enumerate all the printers installed on the machine.
Step2:  For each printer get the handle of the printer by OpenPrinter function.
Step3:  Get the buffer size needed for storing printer information (pointer to a
          variable that the function sets to the size, in bytes, of the printer
          information, and the value represents the required printer buffer size, i.e.
          the value represents the number of bytes stored in the buffer).
Step4:  Get the printer information (pointer to a buffer that receives a structure
          containing information about the specified printer).
Step5:  Get job storage space (required space to printer jobs).
Step6:  Get the number of jobs to be printed.
Step7:  Get the printer information for the Printer Queue and the jobs in the Printer
          Queue.
Step8:  Close the printer.
Step9:  Save the printer information to file.
Step10: If other printer remains Then Go To Step2.
            Else end.
```

2. The scanner monitoring module

The job of this module is to get the scanner information by using **TWAIN** API as mentioned in section (2.7.1). This information is scanner name and status (represent as a number). The scanner status is either switched-ON or switch-OFF.

This module acquires an image from a specific source scanner (need to know exact name of scanner) by using **TWAIN_OpenSource**(Source Name) API, **TWAIN_OpenSource** Opens the Source with the given name. If that source is already opened, does nothing and returns TRUE. If another source is open, closes it and attempts to open the specified source. Or this module acquires an image from the default source by using **TWAIN_OpenDefaultSource** API. **TWAIN_OpenDefaultSource** opens the default source if some source is already open, else does nothing and returns TRUE.

If the open source operation succeeds then acquires an image and stores it in a bmp file, else there is an error in the scanner. To test acquiring an image from the scanner, **TWAIN_AcquireToFilename** API is used, **TWAIN_AcquireToFilename** function is acquire an image and save it to a file. If the filename contains a standard extension (.bmp, .jpg, .jpeg, .tif, .tiff, .png, .pdf, .gif, .dcx) then the file is saved in the implied format.

To hide the source user interface, **TWAIN_SetHideUI** API is used, this function control the 'hide source user interface' flag. This flag is initially false (0), but if set it non-zero, and then when a source is enabled it will be asked to hide its user interface. Algorithm (3.4) shows the steps of the scanner monitoring module:

Algorithm (3.4) scanner monitoring algorithm

Goal: This algorithm monitors the scanner and return scanner information.

Input: None.

Output: Scanner information file.

```
Step1:  Enumerate all the scanners installed on the machine.
Step2:  For each scanner open the source (scanner name).
```

Continued

```
Step3: Acquire image from the default TWAIN device (scanner) and store it to a
           file, then returns a handle to it.
Step4: If previous steps implemented successfully Then save status=0 in the file
          Else save status=1 in the file.
Step5: Close the scanner.
Step6: If  another scanner remains Then Go To Step2.
          Else  end.
```

3. Camera monitoring module

This module gets the camera information by using VfW API as mentioned in section (2.8.1). This information is camera name and status (represent as a number). The camera status is either Connected or Not-Connected.

In this module two APIs of the VfW API are used. The first, **capCreateCaptureWindow** API, is used to find the handle to the specified camera, then create capture window of camera. The second, **capDriverConnect** API, is used to connect a capture window to a capture driver. If this connect is successful then the camera is connected, otherwise there is no camera. Algorithm (3.5) shows these steps.

Algorithm (3.5) camera monitoring algorithm

```
Goal:  This algorithm monitors camera and returns camera information.
Input: None.
Output: Camera information file.

Step1:  Enumerate all the cameras installed on the machine.
Step2:  For each camera get the camera handle.
Step3:  Capture the window of this camera.
Step4:  Capture a camera driver.
Step5:  Connect the captures of window and driver.
```

Continued

```
Step6:  If this connection is true Then save status=0 in the file.
        Else save status=1 in the file.
Step7:  If another camera remains Then Go To Step2.
        Else end.
```

4. Get printer archive module

This function gets the information about a specific printer and stores it on a file. These information are printer name, computer's name that used this printer locally or shared, printed document name, total pages of the printed document, and date and time of printing.

This function opens the specified printer, gets a handle to this printer, then get the number of jobs in the printer's spooler and get the information of each job if printed. Algorithm (3.6) shows the steps of this module.

Algorithm (3.6) Get printer archive algorithm

```
Goal: This algorithm gets the archive of printers.
Input: None.
Output: Printer archive file.

Step1: Enumerate all the printers installed on the machine.
Step2: For each printer get the handle of a specific printer by OpenPrinter function.
Step3: Find out how much memory the archive need.
Step4: Allocate enough memory for used structures.
Step5: Fill out the structure that contains the printer information or job information.
Step6: Get how many jobs in printer spooler.
Step7: Loop through the jobs and access each one
Step8: Save the information about each job in the file.
Step9: Close the printer.
Step10: If another printer remains Then Go To Step2.
Step11: Else end.
```

3.3.2 The Administrator subsystem

The Administrator subsystem is the control unit of the monitoring system. It is loaded on one of the LAN PCs from which the Administrator monitors the LAN. The Administrator selects the PC in the network to monitor its devices by selecting the IP address of the PC. The Agents subsystem in all PCs run in standby mode, ready to receive the Administrator subsystem signal to start running device's monitoring functions and sending the monitoring information. The Administrator is responsible for selecting PC, and receiving the information about the monitored devices.

The Administrator subsystem implemented by using multithreading technique. Threads allow multiple activities to proceed concurrently in the same program. And a multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a *thread,* and each thread defines a separate path of execution. The Administrator subsystem receives each Agent connection. It creates a separate thread to each connection and processes each Agent in a separate thread. These threads enable the Agents to send its' PC name and the flags of devices connection independently and all threads use the same lists to store its name and flags.

Administrator subsystem has four functions, these functions are: **check the Administrator authenticity**, **search for operated (switched ON) computers on LAN, start monitoring, and display printer archive.**

1. Check the Administrator authenticity

This function used to check Administrator authenticity before open the main window of monitoring program. The system gives the user (Administrator) three trails to enter the correct ID and Password. If the user fails, the executed system ends. Algorithm (3.7) used to run this function.

Algorithm (3.7) check Administrator authenticity

```
Goal:  Algorithm is used to check the Administrator authenticity.
Const: Administrator_ID.
       Password.
Input:
       Admin_ID: the name of Administrator.
       Pass: the password of Administrator.
Output: Either true or false.

Step1:  For i=1…3
Step2:      Input Admin_ID and Pass.
Step3:      If  (Administrator_ID = Admin_ID) and (Password = Pass)  Then
                Return (true).
Step4:  Return (false).
```

2. Search for operated (switched-ON) computer on the LAN

This function searches the connected switched-ON computers in the LAN, and the devices connected to each PC. It displays the computers' names and their devices in a list. The administrator can select any computer to monitor its devices, as shown in algorithm (3.8).

Algorithm (3.8) search for switched ON computers on the LAN

```
Goal: This algorithm gets the names of the switched ON computers on the LAN
      and check which devices are connected with these computers.
Input: None.
Output: List of computers' names on LAN and flags of their connected devices.

While (true)
    {
```

Continued

```
    Open server socket with specific port.
    Wait a connection from the Agent program.
    Receive Agents' PC name and put it in PCName_List.
    Receive flag of printer connection.
    Receive flag of scanner connection.
    Receive flag of camera connection.
}
```

3. Start monitoring

This function opens a connection with a selected computer and sends a request to the Agent subsystem which is running on the selected computer. The Agent subsystem collects the information about the computer's device(s) and sends them to the Administrator subsystem. These steps are show in algorithm (3.9).

Algorithm (3.9) Devices monitoring

```
Goal: This algorithm monitors the status of peripheral devices in the Agents' Pc.
Const: Printer_Status list.
Input: PC_Name.
Output: The status of each device in the Agents' PC.

Step1:  Open a connection with Agent by open a socket(PC_Name,  port_No).
Step2:  If flag of printer connection is true Then Receive printer information file.
Step3:  For each printer in printer information file check Printer_Status with
            Printer_Status list to display printer status.
Step4:  If flag of scanner connection is true Then Receive scanner information file.
Step5:  For each scanner in scanne information file check
            If (Scanner_Status = 0)  Then  display the scanner is switched-ON.
            Else display the scanner is switched-OFF.
Step6:  If flag of camera connection is true Then Receive Camera information file.
Step7: For each camera in camera information file
            If (Camera_Status = 0) Then display the camera is Connected.
            Else  display the camera is Not-Connected.
```

4. Display printer archive

This function displays on the administrator screen information about the monitored printer. This can be done by selecting the name of a specific printer to display its' information. These information are recorded by the Agent subsystem on a file on the Administrator computer.

The file of archive contains many information, these information are: **printer name, user name that used this printer, printed documented name, total pages of the printed document, date and time of printing.**

3.4 Software constructing Tools

This project consists of two levels. Each level is responsible for specific jobs. The first (primary) level is responsible for managing the project threads (modules), their creation, running, intercommunication (data transfer) between the threads, and the network connections. It is written using Java language. While, the second (secondary) level is responsible for sending, receiving data from the operating system, and calling operating system instructions. This level is written using Visual C++.

Java language does not support the actions performed in the second level easily and efficiently, while all these actions are found in the Windows API and supported by Visual C++ in an efficient way. So JAVA is used to implement the first level and Visual C++ to implement the second level. To combine java with Visual C++, JNI is used. JNI is found in java assistant programs called JDK 1.3 (Java Development Kits). JNI provides the ability to build the interface between the java and Visual C++ by writing native functions in Visual C++ inside DLL (Dynamic Link Library) file, and then call these functions by the java program to get information from the system. The native functions are used to call the API functions that are found in the Visual C++, as shown in figure (3.4).

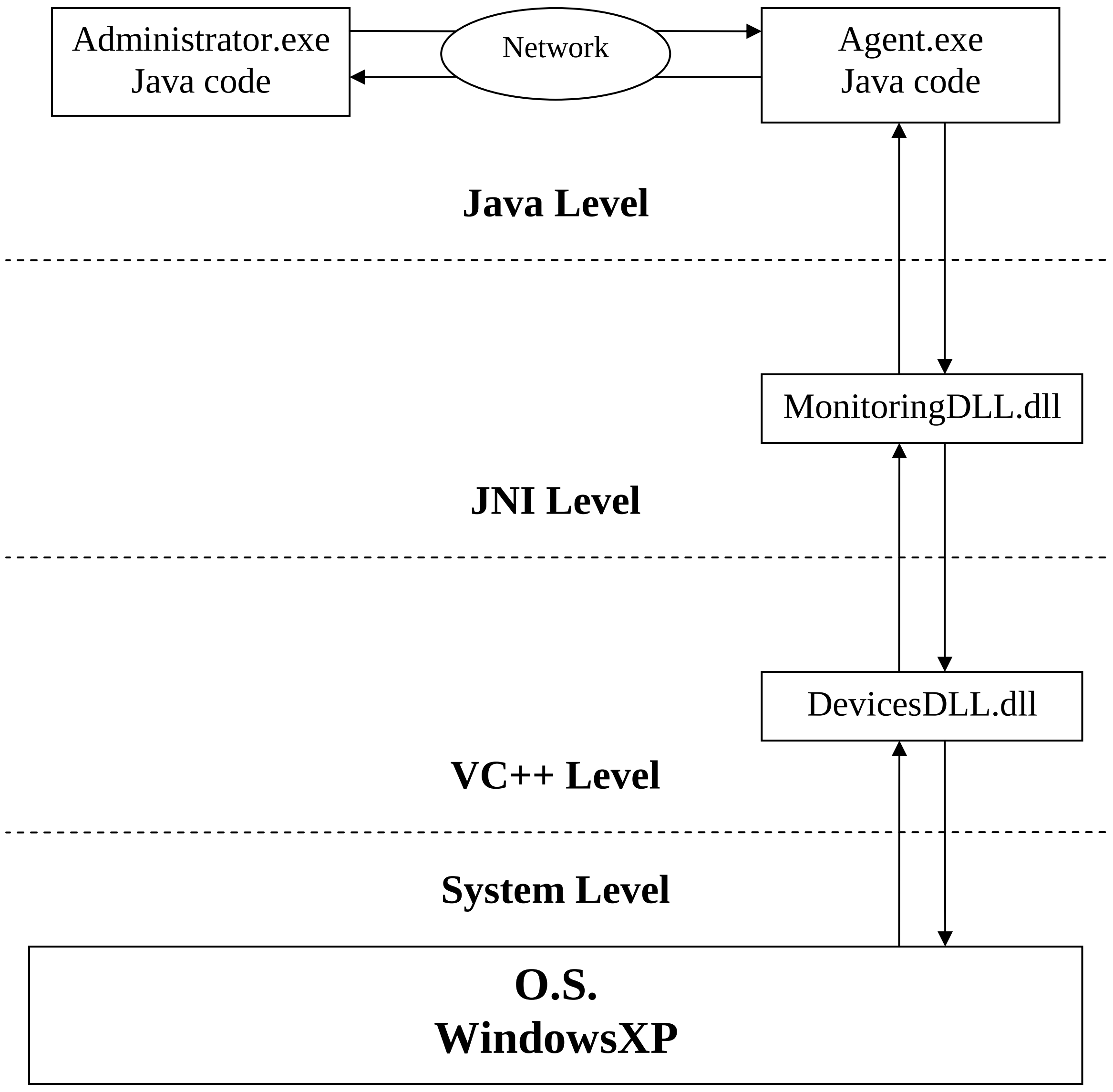


Figure (3.4): The relationship between java and VC++.

To implement DSMS, a DLL file is written using Visual C++. This DLL file contains all native and assistant functions, which are called by the java subsystem of the DSMS. The java subsystem of the DSMS cannot load any DLL library file and use their functions unless this DLL file is created for this java program using JNI methods. So the created DLL file does not represent the function library (in a real meaning) as it is known, it represents an interface between java and Visual C++ programming language.

Chapter two
Computer network and monitoring concept

2.1 Computer Network

The term "computer network" means a collection of autonomous computers interconnected by a single technology. Two computers are said to be interconnected if they are able to exchange information. The connection need to be via a copper wire; fiber optics, microwaves, infrared, and communication satellites. Also computer network is an interconnected system of computing devices that provides shared economical access to computer services. It is a set of computers and peripherals (printers, modems, plotters, scanners, and so on) which are connected together by some medium. The connection can be direct (through a cable) or indirect (through a modem). Different devices on the network communicate with each other through a predefined set of rules (protocols). The device on a network can be in the same room or scattered through a building, or they can even be scattered around the world, connected by a long-distance communication medium [Cra99].

Networks are important since they provide several benefits such as [Tan03]:

- *Resource sharing:* The goal of *Resource Sharing* is to make all programs, equipments, and especially data available to anyone on the network without regard to the physical location of the resource and the user. Files, programs, printers and other devices can all be shared among many users, reducing the cost per user. For example, having a group of office workers share a common printer, none of the individuals really needs a private printer, and a high-volume networked printer is often cheaper, faster, and easier to maintain than a large collection of individual printers.

- *Increased reliability:* rather than relying on a central computer for storage and processing, many computers are available. If the user's computer fails, he can login to another computer and access the same file and programs from the server.
- *Better price/performance ratios than centralized systems:* generally, the entire network of workstations and servers, including the network hardware, is less expensive than a mainframe computer and, for each individual user, performs as well. This is due to the rapidly decreasing cost of personal computers and related hardware.
- *Distributed systems vs. mainframes:* distributed systems offer performance that rivals mainframe systems at a fraction of the cost.
- *Improved person-to-person communications*: businesses and individuals can take advantage of the speed and low cost of long distance communications that WANs, such as the Internet, provide. For example: compare the cost of sending several employees to a meeting that is across the country, to a video conference via the internet.

There are two main types of network connection: **client/server** and **peer-to-peer** [Cra99, Pre05].

1. **Client/server** describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfills the request. Although the client/server idea can be used by programs within a single computer, it is a more important idea in a network. In a network, the client/server model

provides a convenient way to interconnect programs that are distributed efficiently across different locations. Computer transactions using the client/server model are very common. For example, to check the bank account of any person from remote computer, a client program in remote computer forwards request to a server program at the bank. That program may in turn forward the request to its own client program that sends a request to a database server at another bank computer to retrieve the account balance. The balance is returned back to the bank data client, which in turn serves it back to the client in remote personal computer, which displays the information for this person.

Another definition of client/server is a computational architecture that involves client processes requesting service from server processes. In general, client/server maintains a distinction between processes and network devices. Usually a client computer and a server computer are two separate devices, each customized for their designed purpose. For example, a Web server will often contain large amounts of memory and disk space, whereas Web clients often include features to support the graphic user interface of the browser such as high-end video cards and large-screen displays. Figure (2.1) illustrates the logical architecture of a typical client/server network [Cra99, Pre05].

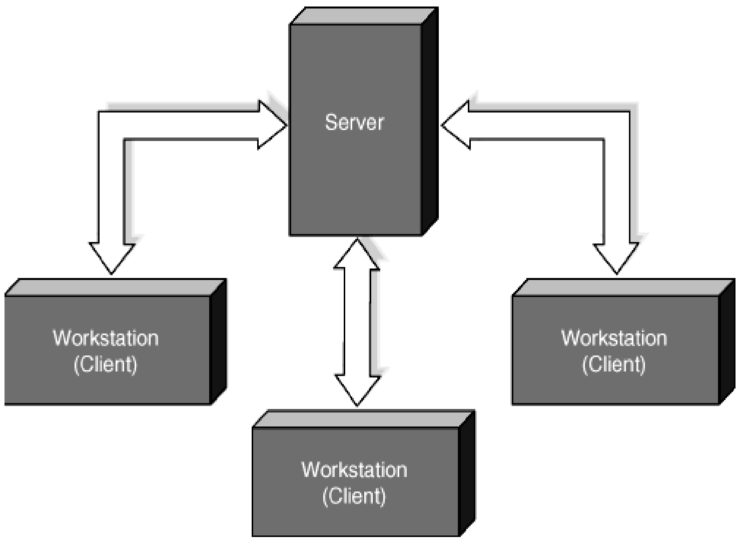


Figure (2.1): The logical architecture of a typical client/server network.

2. **Peer-to-peer** networks have workstations connected to each other but do not have servers. This configuration generally has a single string of computers connected together via cabling. Each computer is an equal, or "peer," of the others, and it can share the files and peripherals of other computers connected to the network [Cra99].

If one user turns off his/her workstation, his/her information or peripherals will no longer be available for others to share. In addition, accessing data and applications from another person's workstation can cause performance problems for that user. Peer-to-peer networks are typically used for connecting nodes via largely connections. Such networks are useful for many purposes. Sharing content files containing audio, video, data or anything in digital format is very common, and real-time data, such as telephony traffic, is also passed using peer-to-peer technology. An important goal in peer-to-peer networks is that all clients provide resources, including bandwidth, storage space, and computing power. Thus, as nodes arrive and demand on the system increases, the total capacity of the system also increases. This is not true of client-server architecture with a fixed set of servers, in which adding more clients could mean slower data transfer for all users. Figure (2.2) illustrates peer-to-peer model [Cra99].

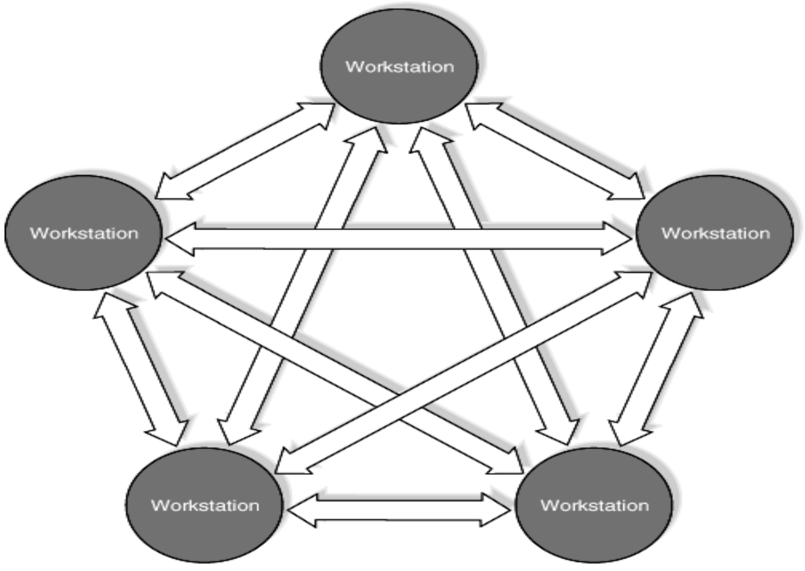


Figure (2.2): Peer-to-peer models.

2.2 Network hardware

Networks can be classified using their scale. Figure (2.3) shows the classification of multiple processor systems arranged by their physical size. At the top are **data flow machines,** highly parallel computers with many functional units all working on the same program. Next are the **multi-computers**, systems that communicate by sending messages over very short, very fast buses. Computers that communicate by exchanging messages over longer cables, these can be divided in to **local, metropolitan, wide area networks.** Finally, the connection of two or more networks is called an **internet** work [Tan03].

Interprocessor Distance	Processors Located in same
0.1 m	Circuit board
1 m	System
10 m	Room
100 m	Building
1 km	Campus
10 km	City
100 km	Country
1000 km	Continent
10000 km	Planet

Data flow machine

Multi-computers

Local Area Network

Metropolitan Area Network

Wide Area Network

Internet

Figure (2.3): Classification of Interconnected Processors by scale.

- **Local Area Network (LAN):** LANs are privately-owned networks within a single building or campus of up to a few kilometers in size. They are widely used to connect personal computers and wo company offices and factories to share resources (e.g., printers) and exchange information. LANs are distinguished from other kinds of networks by three characteristics: (1) their size, (2) their transmission technology, and (3) their topology.

LANs are restricted in size, which means that the worst-case transmission time is bounded and known in advance.

LANs may use a transmission technology consisting of a cable to which all the machines are attached, like the telephone company party lines once used in rural areas.

Various topologies are possible for **LANs**; these topologies are ring, star, and bus or tree. [Tan03].

- **Metropolitan Area Network (MAN):** MAN covers a city. The best-known example of a MAN is the cable television network available in many cities. This system grew from earlier community antenna systems used in areas with poor over-the-air television reception. In these early systems, a large antenna was placed on top of a nearly hill and signal was then piped to the subscribers' houses. MANs often enable users in several local geographical locations to use shared network resources as if they were all parts of the same local network [Mat98, Tan03].

- **Wide Area Network (WAN):** WAN spans a large geographical area, often a country or continent. It contains a collection of machines intended for running user (i.e., application) programs. These machines traditionally called hosts**.** At a basic level, a WAN can be created by tying a series of simple, point-to-point links together. On the other end of the spectrum, a WAN might comprise many different systems and LANs, all interconnected using a variety of techniques, including standard telephone lines, packet-switching networks and Integrated Services Data Network (ISDN) links. Between the two extremes are networks that are superficially simple but technically complex, and those that are superficially complex but technically simple [Dan99, Tan03].

2.3 Network software

The first computer networks were designed with the hardware as the main concern and the software as an afterthought. This strategy no longer works. Network software is now highly structured. It can be described from two points of views, from protocol point of view and from InterProcess

Communications (IPC) point of view [Tan03].

2.3.1 Network protocols

A protocol is a set of rules and specifications governing how two entities communicate. Computers have certain defined protocols specifying proper behavior for communication between them. When any hardware or software violates these rules, proper communications may not be able to take place over the network, even between other systems that are following the rules. Messages generated by a machine not conforming to accepted protocols probably won't be acknowledged by other computer [Cra99].

It is helpful to distinguish some different types of protocols. Understanding these different types is where a construction such as the ***Open System Interconnection*** (**OSI**) and ***Transmission Control Protocol/Internet Protocol*** **TCP/IP** protocols network model comes in handy. The **OSI** model is a construction that has seven basic levels of network communication, ranging from the physical medium of the network all the way up to the application interface appearing on the workstation. The entire **OSI** model is not implemented, where the most common layered set of protocols in use is the **TCP/IP**.

The most common example of the network protocols is the (**TCP/IP**) that map to a four-layered conceptual model: Application, Transport, Internet, and Network Interface. As shown in figure (2.4), each layer in the TCP/IP model corresponding to one or more layers of the seven-layers of (OSI) model. TCP/IP is a broad set of rules and standards, these rules control how data on the network is sent on the correct path to reach its intended destination, how some communication errors are handled, and how logical connections between nodes on the network are established, maintained, and ended [Cra99, Fai00].

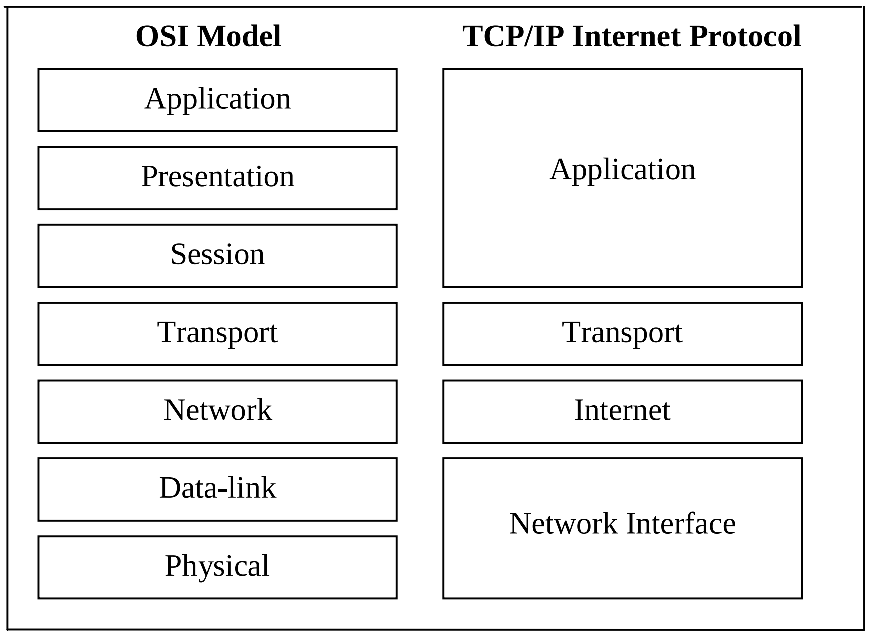


Figure (2.4): TCP/IP and the OSI Model.

2.3.2 InterProcess Communication (IPC)

The connection between the client and server portions must allow data to flow in both directions. There are number of ways to establish this connection. Operating Systems (OSs) support mechanisms of IPC. Many of these mechanisms are similar in its function but different in name from one OS to another. As an example Windows NT operating system provides several different IPC mechanisms. The most common Windows IPC are [Cou01, Dei01, Lit00, Par99, Pra97]:

1. **Remote Procedure Calls (RPCs):** The RPC was designed as a way to abstract the procedure-call mechanism for use between systems with network connections. In RPC the client program calls a procedure in

another program running in server process. Servers may be clients to other servers to allow chains in RPCs. RPC allows a procedural program to call a function residing on another computer as conveniently as if that function where part of the same program running on the same computer. A disadvantage of RPC is that it supports a limited set of simple data types. Therefore, RPC is not suitable for passing and returning java objects.

2. **Sockets:** All upper-layer applications that uses TCP have a ***port*** number those identifiers the application. In theory, port numbers can be assigned on individual machines however the administrator desires, but some conventions have been adopted to enable better communications between TCP implementations, which enables the port number to identify the type of service that one TCP system is requesting from another (for example "Simple Mail Transfer Protocol" SMTP service available on port 25). Port numbers can be changed, although this can cause difficulties. Most systems maintain a file of port numbers and their corresponding service. Each communication circuit into and out of the TCP layer is uniquely identified by a combination of two numbers, which together are called a ***socket.*** The socket is a mechanism between network applications running on the same computer, or on different computers connected using a LAN or WAN. It defines a set of standard Application Program Interface (APIs) that an application uses to communicate with one or more other applications, usually across a network. The socket supports:

i. Initiating an outbound connection for a client application.

ii. Accepting an inbound connection for server application.

iii. Sending and receiving data on a client/server connection.

iv. Terminating a client/server connection.

The specification includes a standard set of APIs supported by all Windows-based TCP/IP protocol stack, and to be used by network applications. In Sockets, application communications channels are represented by data structures called *sockets*. Two items are used to identify *a socket*:

- An IP address.
- A port number.

Both the sending and receiving machines have sockets. Because the IP address is unique across the inter network, and the port numbers are unique to the individual machine, the socket numbers are also unique across the entire inter network. This enables a process to talk to another process across the network, based entirely on the socket number.

Sockets are highly useful in at least three communications contexts:

a. Client/Server models.

b. Peer-to-Peer scenarios, such as chat applications.

c. Making remote procedure calls (RPC) by having the receiving application interpret a message as a function call.

3. **Remote Method Invocation (RMI):** RMI is a Java's implementation of RPC for java-object-to-java-object distributed communication. Sockets are built around sending bytes while RMI provides a way to call methods on objects on other systems. It operates at a higher level of abstraction than socket-based programming, even though RMI does use sockets under the covers. RMI enables the programmer to create distributed Java-to-Java applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. A Java program can make a call on a remote object once it obtains a reference to the remote object, either by looking up the remote object in the bootstrap naming service provided by RMI or by receiving the reference as an argument or a return value. A client can call a remote object in a server, and that server can also be a client of other remote objects. RMI uses object serialization to marshal and unmarshall parameters and does not truncate types, supporting true object-oriented polymorphism.

2.4 Network monitoring

Network monitoring software is software that is designed to help monitor LANs, MANs, WANs, and all network equipment components. With network monitoring software implemented, user will get alerts when network equipment fails, monitor performance of network equipment, track and troubleshoot network related issues, provide reports about the network and equipment, and much more.

Network monitoring software can be very helpful when trying to troubleshoot network related issues, watching network equipment and providing performance analysis. Network monitoring software is a must have if the user is a network administrator, it also can allow to monitor a single network device or all network devices. It is concerned with observing and analyzing the status and behavior of the managed objects (end systems, intermediate systems, and sub networks). Network monitoring consists of three major design areas [Led01, Lui99]:

- Access to monitored information: how to define monitoring information, and how to get that information from a resource to a manager.
- Design of monitoring mechanisms: how best to obtain information from resources.
- Application of monitored information: how the monitored information is used in various management functional areas.

Network monitoring Focus on three functional areas: Performance, fault and accounting monitoring.

2.4.1 Performance monitoring

Performance monitoring deals with measuring the performance of the network. There are three important issues in performance monitoring. First, performance monitoring information is usually used to plan future network expansion and locate current network usage problems. Second, the time frame of performance monitoring must be long enough to establish a network behavior model. Third, choosing what to measure is important. There are too many measurable things in a network. But the list of items to be measured should be meaningful and cost effective. This list of items to be measured is called network indicators because they indicate attributes of the network [Edm00].

2.4.2 Fault monitoring

Fault monitoring is a simple but pragmatic approach to network monitoring, as it focuses on the state of the various network elements only, i.e. whether an element is operational and whether it is accomplishing the services and tasks that are assigned to it. It doesn't provide any information concerning the performance or activity of the network and its nodes. Service faults are detected instantly whereupon the operator is alerted.

The objective of fault monitoring is to identify faults as quickly as possible after they occur and to identify the cause of the fault so that remedial action may be taken. The Problems of fault monitoring are [Led01]:

- **Unobservable faults:** certain faults cannot be observed locally (e.g. deadlocks), and others may not be observed because the vendor equipment is not instrumented to record the occurrence of a fault.
- **Partially observable faults**: A node failure is observable, but this may be insufficient to pinpoint the problem (e.g. failure of a low-level protocol in an attached device).
- **Uncertainty in observation**: Lack of response from a remote device may mean that the device is stuck or the network is partitioned, or congestion causes delays.

2.4.3 Accounting monitoring

Account monitoring deals with how users use the network. The network keeps a record of what devices of the network are used by users and how often they are used. This type of information is used for billing user for network usage, and for predicting future network usage.

Accounted resources can be: communication facilities, computer hardware, services, and Accounting data can include user identification, receiver, number of packets, resource used [Led01].

2.5 Peripheral devices

A peripheral device is any part of a computer other than the Motherboard, CPU or working memory. It is a device that operates in combination or conjunction with the computer but is not physically part of the computer and is not essential to the basic operation of the system, for example, disks, keyboards, monitors, mice, printers, scanners, tape drives, microphones, speakers, cameras, etc. The peripheral device consists of two parts: the peripheral device hardware (Device Controller), and the peripheral device software (Device Driver).

A computer can use a network to access peripheral devices. For example, all computers on a network can access a printer attached to the network. Similarly, all computers on a network can share files on a disk that is attached to the network [Dou99].

A communications network connects numerous peripheral devices to a host computer via a single host interface (device driver). The host interface and each peripheral device's interface have its own CPU, with software for assigning each peripheral device a unique address. The bus interface associated with each peripheral device typically stores a unique identifier string that is used by the host computer to identify each peripheral device connected to the network. Alternately, the host can distinguish identical peripheral devices by the order in which they are first used. As a result, several peripheral devices of the same type can be connected to the network, each being assigned a distinct network address. Peripherals can be connected and disconnected to the desktop bus while the system is running [Fre04, Wik06].

2.5.1 Peripheral devices hardware (Device controller)

I/O devices or peripheral devices typically consist of a mechanical component and an electronic component. The electronic component is called the **device controller** or adapter. On personal computers, it often takes the form of a printed circuit card that can be inserted into a slot on the computer's parentboard or motherboard. The mechanical component is the device itself.

The controller card usually has a connector on it, into which a cable leading to the device itself can be plugged. Many controllers can handle two, four, or even eight identical devices. If the interface between the controller and device is a standard interface, then companies can make controllers or devices that fit that interface. Many companies, for example, make disk drives that match the Integrated Drive Electronics (IDE) or Small Computer System Interface (SCSI) disk controller interfaces. The operating system nearly always deals with the controller, not the device. And the interface between the controller and the device is often a very low-level interface.

Each controller has a few registers that are used for communicating with the CPU. On some computers, these registers are part of the regular memory address space [Wik06, Woo97].

2.5.2 Peripheral devices software (Device Driver)

Each device driver handles one device type, or at most, one class of closely related devices. Each controller has one or more device registers used to give it commands. The device drivers issue these commands and check that they are carried out properly.

In general terms, the job of a device driver is to accept abstract requests from the device-independent software and see to it that the request is executed, it must decide which controller operations are required and in what sequence.

Once it has determined which commands to issue to the controller, it starts issuing them by writing into the controller's device registers. Some controllers can handle only one command at a time. Other controllers are willing to accept a linked list of commands, which they then carry out by themselves without further help from the operating system. In many cases the device driver must wait until the controller does some work for it, so it blocks itself until the interrupt comes in to unblock it. A part from the device-driver layer supervises [Woo97]:

- Management of the name space for files and devices.
- Access control to files and devices.
- Operation control.
- File system space allocation.
- Device allocation.
- Buffering, caching, and spooling.
- I/O scheduling.
- Device status monitoring, error handling, and failure recovery.
- Device driver configuration and initialization.

2.6 Printer

A printer is a device that interprets digital information and produces the actual print by applying inks on a piece of paper. The printed content comes from a computer system like a desktop computer, for example. Usually an end user initiates the printing process through some graphical user interface. Of course, an automated software process could start the process also.

The printer represents the actual physical device that produces the actual print. Printers are typically connected with PCs. Older models might rely on a serial or parallel port, whereas I/O port in the latest models usually USB (Universal Serial Bus). More advanced models may be equipped also with for example Ethernet, WLAN (Wireless LAN) or Bluetooth support [Tim05].

A Printer, as referred to by the Win32 API, is conceptually an entity comprising the printer driver, the print queue, and the input/output path to the physical printer. The operating system treats a physical printer as merely the destination of a print job generated by and passed through a system "Printer", referred to in this thesis.

The most visible part of a Printer is a print queue. It is managed by the Print Manager or the Printer folders in the Windows 95-style user interfaces. The printer driver is the interface to the Printer that is used by applications to create print jobs via printer. The I/O path for a Printer consists of several layers of system code culminating with a port monitor.

The port monitor is the interface to the physical printer at the down-stream end of a system Printer and is responsible for transferring the data of a print job across whatever connection exists to the physical printer. In the case of bi-directional printers, the port monitor would be responsible for transferring data to and from the physical printer. This connection, and the physical printer, is where errors occur. It is the job of the port monitor to report those errors.

The Spooler does not query for the state of a physical printer to which a Printer is connected. Instead, the state of a physical printer determines the success of a print job at the time it is despooled over the port monitor. If some error occurs in this process, the error is reported by the port monitor and recorded in a print job's status information. The Spooler, in turn, propagates reasonable error information to the Printer Queue.

Consequently, a system Printer reports no status when the Printer queue is empty. In this state, the Printer is assumed ready to accept print jobs. This is a valid assumption even if the physical printer is in an error state such as off-line. The operating system considers the Printer ready to accept print jobs even if, for some reason, it cannot complete delivery to the physical printer. Such a circumstance is considered an error state in the operating system that must be addressed by the user. It is not considered an error reportable to the application that is allowed to complete the spooling of the print job successfully [Mho00].

- **Printing system**

Printing is a process where an image-text, graphics or both- is produced either on paper or on some other 2-diemensional surface.

The term printing system is referred to everything between the client and the physical printer device. A typical system is a software module that handles some common tasks involved in printing. Printing systems try to fulfill the needs of the clients, while the must adhere to the restrictions set by the printer device. The purpose of a printing system is to make the printing easier for the clients. In principle a client could be directly connected with a printer, but in that case the client with a higher-level interface, which is more straightforward to use and hides the differences between printer devices.

Typically a printing system has multiple clients. Printing is usually a service offered by an operating system and all the applications running on the platform may use the same services.

The printing system may consist of multiple software processes that might be hosted on different computers. On a PC that is directly connected to a printer the printing system could refer to the printing capabilities provided by the operating system and would be situated on a single computer. If the printer would be a network printer, the printing system could consist of multiple computers. One part of the system would be on the same computer as the client process and respond to the print requests of the client. The requests would then be forwarded over the network to a component on another computer that would actually be connected with a printer [App02, Tim05].

At least part of a printing system relies typically on the same system as the clients, because then it can communicate with the clients through API.

Printing systems need to be compatible with various kinds of printers. For this a printing system typically uses a printer driver for controlling a printer device. Printer driver is usually a small application that can control and feed data to a particular kind of printer, a printer controller (or adapter) that makes a printing system and a printer work together. Drivers are typically customized for a certain printing system and a certain printer. Printer venders may provide several drivers that make it possible to use their printers on the most popular platforms.

Printer driver is configured during the printer installion. On PCs, the driver may be provided, e.g., on a floppy disc or CD-ROM or fetched from the internet. Also, because hard disc space is usually abundant, the operating system may readily contain the printer drivers for the most popular printer models.

In addition to submitting print jobs to the printer, the printing system needs to be aware of the status of the printer. For example, a printer may be ready to accept new jobs, be busy or be in some error condition that requires human intervention [Tim05].

2.7 Scanner

It is an input device used to translate a picture or typed text into a pattern of dots which can be understood and stored by a computer. Scanner is a piece of hardware used to scan a document, i.e., creates a digital copy. Although flatbed scanners are the most common type**.**

It is typically connect to a computer using one of four methods: parallel port, USB, SCSI, or IEEE 1394. In this project, the scanner connects to computer using USB port.

A scanner with a USB interface is as easy to attach to the computer as a parallel port scanner; simply plug the USB cable into one of the USB ports on the computer. USB connections operate significantly faster than parallel port connections. The scanner has to be communicating with the user application. The method which is used in this project to connect scanner and user application is TWAIN API [Mim00].

2.7.1 TWAIN API

In the past, development of a new scanner inevitably required the development of a compatible driver and a sample (demonstration) program exclusively designed for the new scanner. As the scanner is upgraded, the driver may need to be upgraded as well to maintain its compatibility with the upgraded scanner. Therefore, the use is most likely to be restricted to one specific scanner model to avoid the complication of learning new operation methods and replacing the peripheral driver, etc., involved in scanner replacement. In view of the incompatibility among different scanners and peripheral equipment and the accompanying inconvenience, there was a demand for standardization of the related hardware and software, and TWAIN was established as a result.

TWAIN defines a standard software protocol and API for communication between software applications and image acquisition devices, and provides easy integration of image data between input devices, such as scanners and digital cameras, and software applications.

The TWAIN initiative was originally launched in 1992 by leading industry vendors who recognized a need for a standard software protocol and API that regulates communication between software applications and imaging devices (the source of the data), TWAIN defines that standard [Fuj01, Twa06].

With TWAIN, an application program can acquire images through any device that complies with its specification. Any scanner, or other image acquisition device, to function properly, needs the TWAIN data source from its manufacturer as well as the TWAIN DLLs. The required TWAIN DLLs are provided with Imaging.

TWAIN was designed to provide a consistent, easy integration of image data between sophisticated input devices and software applications. The goals of TWAIN are [Twa06]:

- Multiple platform support - The interface must work across many operating systems. The use of platform-specific mechanisms should be kept to a minimum.
- Support for multiple devices - These include hand-held scanners, flatbed scanners, slide scanners, image capture boards or frame grabbers, digital cameras and image databases-a broad range of raster image- generating devices.
- Widespread acceptance - Provide an interface that is well-defined and enables the majority of the leading hardware and software developers to provide drivers for their devices or include support through their applications.
- Extensibility and revisions - As the industry grows, the specification's architecture should be extensible and able to handle new, unknown conditions. Revisions will be backwards-compatible with code written to earlier versions of the specification.
- Easy implementation - The interface and its documentation should be clear, well structured, well written, and intuitively designed for developers to learn and write the code for it.

2.7.2 Communicating with TWAIN

TWAIN enables users to acquire images directly from an input device. Hardware vendors can write a single TWAIN driver for a device, which can then be used by all TWAIN applications to acquire images. Figure (2.5) presents a simple overview of the relationship between input devices and applications using TWAIN.

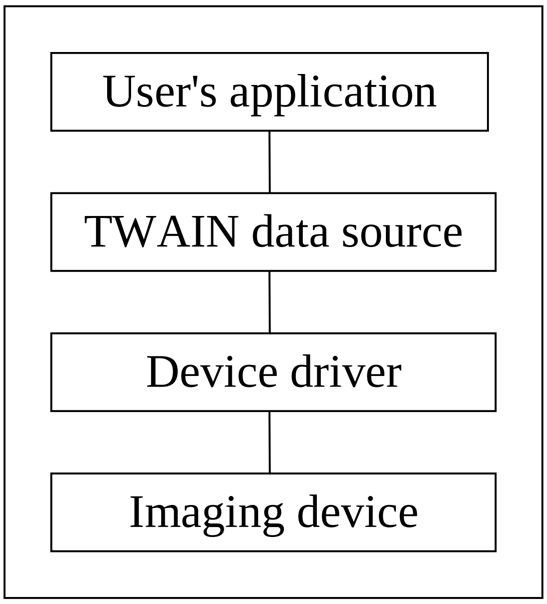


Figure (2.5): TWAIN architectural overview.

Figure (2.5) uses a four-layer protocol to illustrate how applications and input devices communicate using TWAIN. Table (2.1) outlines the four-layer protocol and its function [Mim00].

Table (2.1) TWAIN architectural layer functions

Architectural layer	**Function**
User's application	Represents the user's application, used to scan or otherwise acquire an image from a TWAIN input device, such as a scanner or camera.. The application requesting an image sends the request to the TWAIN data source.
TWAIN data source	Implements instructions and communications required for transferring the image.
Device driver	Represents the input device. The software elements that control acquisitions, called data sources, reside in this layer. A data source is typically a TWAIN driver written by a hardware vendor to get data from a hardware device and pass it to a TWAIN application.
imaging device	Represents a physical input device, such as a scanner or digital camera.

2.8 Camera

Camera is a video capture device. The video capture device is a source of video information placed in the frame buffer. The video source might be a video camera, video player, or television tuner. The format of both the incoming signal and the data placed in the frame buffer is controlled by the video capture hardware. The way which used to connect between the application and camera device is Video for Windows (VfW) API [Mvi00].

- **Video for Windows (VfW)**

VfW was a multimedia technology developed by Microsoft that allowed Microsoft Windows to play digital video. VfW is the first video capture and display system developed by Microsoft for the Windows operating system. The design of VfW video capture was optimized for capturing movies to disk. Features important to video conferencing, TV viewing, capture of video fields, and ancillary data streams are missing from the VfW architecture. To circumvent these limitations, vendors augmented VfW by implementing proprietary extensions. However, without standardized interfaces, applications that use these features must include hardware-dependent code.

There were three components in VfW. The technology introduced a file format designed to store digital video, Audio Video Interleave (AVI). The technology provided an application programming interface that allowed software developers working on the Windows platform to add the ability to play or manipulate digital video to their own applications. Lastly, it includes a suite of software for playing and manipulating digital video [Ans06, Wkp06].

2.9 Java Native Interface (JNI)

JNI is the native programming interface for java that is part of the java. The JNI allows java code that runs within a Java Virtual Machine (JVM) to operate with applications and libraries written in other languages, such as C, C++, and assembly programmers use the JNI to write native methods to handle those situations when an application cannot be written entirely in the java programming language. It may need to use native methods and the JNI in the following situations [Jav05]:

- The standard java class library may not support the platform-dependent features needed by the application.
- There exists a library or application written in another programming language which it needs to be accessible by java applications.
- A small portion of time-critical code needs to be implemented in a lower-level programming language, such as assembly, and then called by java application.

Programming through the JNI framework lets the user use native methods to do many operations. Native methods may be written explicitly to solve a problem that is best handled outside of the java programming environment. The JNI framework lets the native method utilize java objects in the same way that java code uses these objects. A native method can create java objects, including arrays and strings, and then inspect and use these objects to perform its task. A native method can also inspect and use objects created by java application code. A native method can even update java objects that it created or that were passed to it, and these updated objects are available to the java application. Thus, both the native language side and the java side of an application can create, update, and access java objects and then share these objects between them.

Writing native methods for Java programs is a multi-step process. The programmer has to perform the following steps [Jav05]:

1. Begin by writing the Java program. Create a Java class that declares the native method; this class contains the declaration or signature for the native method. It also includes a main method which calls the native method.

2. Compile the Java class that declares the native method and the main method.
3. Generate a header file for the native method using javah with the native interface flag -jni. Once the header file is generated the formal signature for the native method is defined.
4. Write the implementation of the native method in the programming language of the programmer choice, such as C or C++.
5. Compile the header and implementation files into a shared library file.
6. Run the Java program.

Figure (2.6) illustrates the above steps.

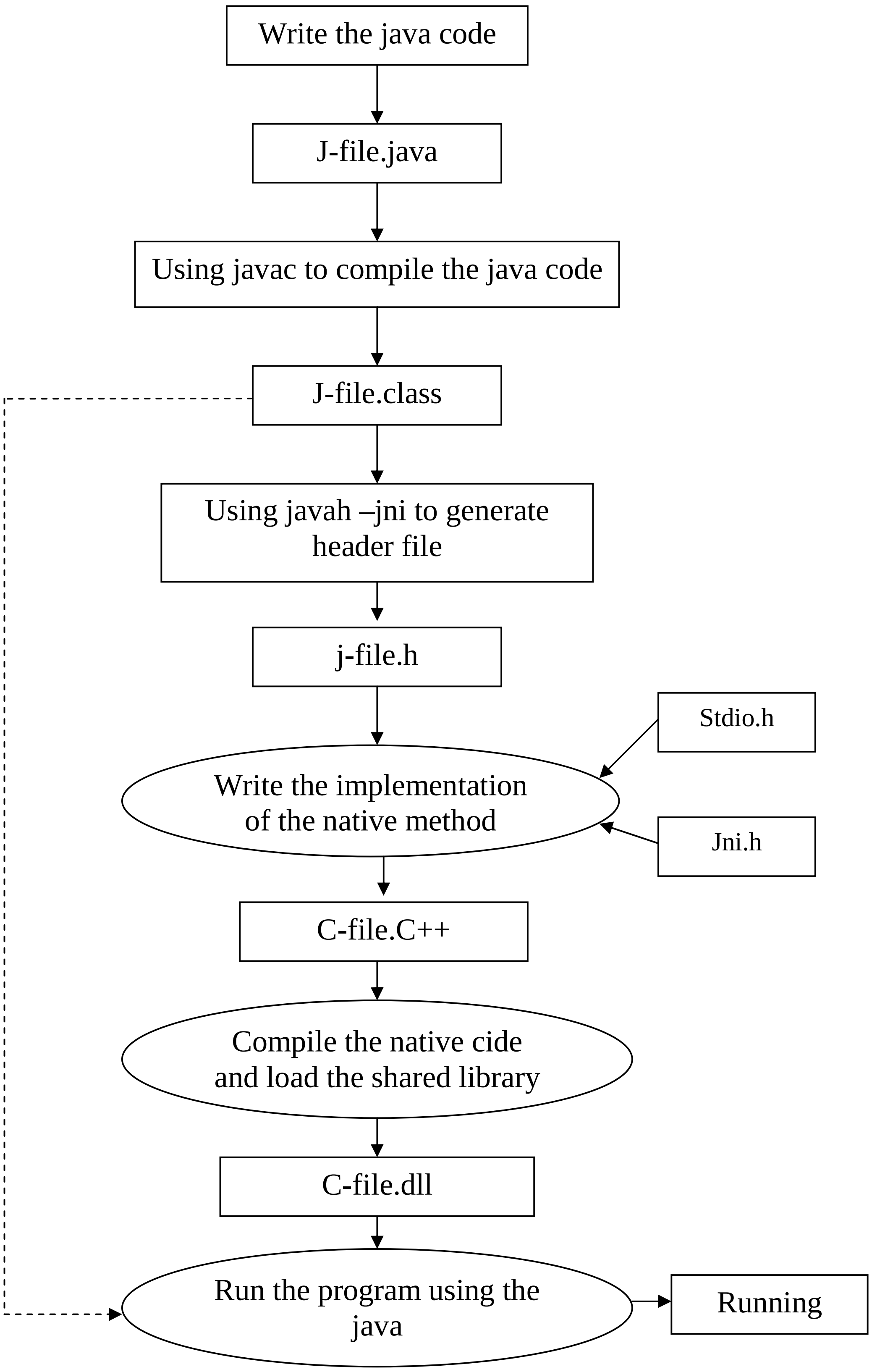

Write the java code
J-file.java
Using javac to compile the java code
J-file.class
Using javah –jni to generate header file
j-file.h
Stdio.h
Write the implementation of the native method
Jni.h
C-file.C++
Compile the native cide and load the shared library
C-file.dll
Run the program using the java
Running

Figure (2.6): Steps followed to write native method.

(- - - - - -▸ means run the class)