The Renaissance of Distributed Computing

White Paper
March 1997 Systems Design Review

©1996, 1997 Microsoft Corporation. All rights reserved.
This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Microsoft, Windows, Windows NT, Visual Basic, and BackOffice are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
Other products or company names mentioned herein may be the trademarks of their respective owners.
Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA

ABSTRACT

This paper describes key new elements of Microsoft's platform technologies—how they benefit the user and the network administrator, and how they will simplify development of distributed applications. In particular, it focuses on the problems associated with computing in a networked environment, and how the Microsoft® Windows NT® operating system and related technologies are solving these problems.

Table of Contents

INTRODUCTION 1

THE PROMISE OF CLIENT–SERVER COMPUTING 2

WINDOWS 95 AND WINDOWS NT: INTEGRATING THE NETWORK AND THE OPERATING SYSTEM 2

ROADBLOCKS TO DISTRIBUTED COMPUTING 3

MAKING IT EASIER TO USE AND MANAGE THE NETWORK 4

THE CHALLENGES OF PROGRAMMING IN A DISTRIBUTED ENVIRONMENT 9

THE INTERNET'S IMPACT ON DISTRIBUTED COMPUTING 10

THE ACTIVE SERVER – MAKING IT EASIER TO PROGRAM IN A DISTRIBUTED ENVIRONMENT 10

WHAT THE FUTURE HOLDS 13

For more information 13

The Renaissance of Distributed Computing

Introduction

In their efforts to increase online efficiency while reducing development and training costs, businesses have demanded a common solution for storing, publishing, and retrieving information, and for running line of business applications that will work on both internal networks and on the Internet. Microsoft's response is an architecture that is based on a symmetric set of building blocks that include HTML, scripting, components, and system services. The native, comprehensive, high performance implementation of this platform uses Microsoft® Windows® 95 or Microsoft Windows NT® on the client and Windows NT on the server.

Microsoft Windows offers users and administrators a graphically rich, intuitive, easy-to-manage, and consistent experience. Developers benefit from rich services built into the system and a uniform programming model across the client and the server that they can access from a consistent, integrated set of tools.

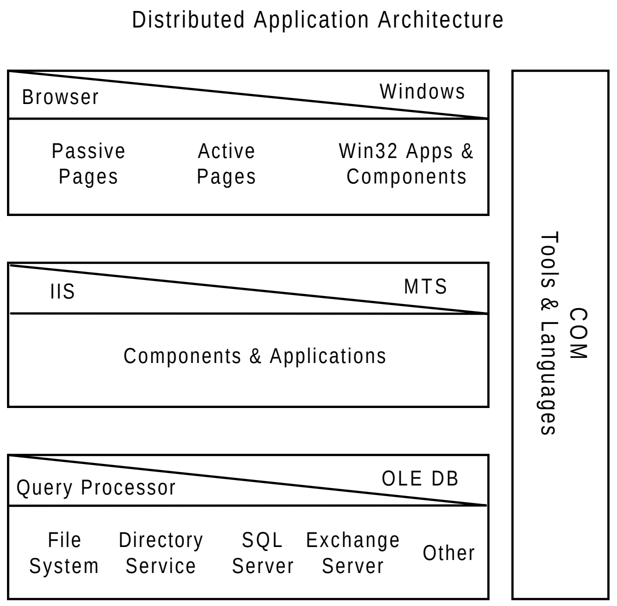


Figure 1. Microsoft's development platform brings information to the desktop from the Internet and intranet through rich system services, industry standard protocols, and a User Interface that is web-based or a traditional Win32 application.

The Internet has created a single paradigm for accessing information—a browser that interprets a standard language (HTML) and understands standard protocols. Microsoft Internet Explorer "embraces and extends" this universal client with scripting and components that support dynamic, animated content. The next major releases of Windows – "Memphis" and Windows NT 5.0 – will include its next incarnation, Internet Explorer 4.0, which combines the best of the browser with the best of the Windows User Interface. It gives users the choice of navigating their hard drive and network the same way they navigate the Web – with single mouse clicks, hot-links, and a customizable desktop that looks and behaves like an HTML page.

While the native implementation of Microsoft's client technologies is for Windows 9x and Windows NT, it is specifically designed to support the development of client software that can run on other operating systems, including the Macintosh and UNIX. Windows NT Server and related services (see figure 1) are the cornerstone of the infrastructure required to store, process, manage, and deliver information to clients.

It creates an easy-to-use framework that enables a wider range of developers to write server applications for the intranet and Internet, because it handles the complex tasks traditionally involved with server programming.

With the aid of powerful development tools, Microsoft's development platform is enabling an explosion in distributed applications that are comprehensive, simple-to-create, and that are tightly integrated with operating system services and Intranet standards.

This paper focuses on the key new elements of Microsoft's platform technologies—how they benefit the user and the network administrator, and how they will simplify development of distributed applications. In particular, it focuses on the problems associated with computing in a networked environment, and how Windows NT and related server technologies are solving these problems.

The Promise of Client–Server Computing

The computing industry has long hailed client-server computing as a promising alternative to mainframes. The idea was that inexpensive servers would replace the centralized model of mainframe computing, particularly in building line of business solutions. Mainframes were, in a word, expensive – to buy, to install, and to maintain. A single machine meant a single point of failure, and requiring that users always share resources, even if they were doing unrelated tasks, meant that to add more users (to "scale") you eventually had to replace your mainframe with an even bigger (and more expensive) machine.

As client-server gained momentum, however, it introduced a new set of costs. Instead of working together as a single unit, servers were actually deployed like individual, small-scale mainframes; as a result, the more servers there were, the harder it was to manage a network. Writing applications that insulated users and administrators from dealing with multiple servers was difficult, in particular because the operating system and network services were not completely integrated. Users looking for a specific piece of information had to know on which machine it was physically stored. In reality, the user and administrator experience under the client-server model became more complicated than it had been under the centralized mainframe model.

Even so, the transition to client-server produced many benefits – in particular, efficiency gains and increased flexibility in the use of resources. For example, client-server broke the management load into smaller pieces. Users had their own desktop machines for independent tasks like writing letters that didn't burden shared computing resources the way they did on the mainframe. Yet users could access shared resources, like printers, file servers, electronic mail, and databases, through servers on the network.

Windows 95 and Windows NT: Integrating the Network and the Operating System

The client/server model drove the first steps taken to integrate the network into the operating system. It motivated the designers of Windows and Windows NT to solve many important problems associated with networked PCs, for example:

- **Managing security on servers is logically centralized, which greatly simplifies administration**. Many administrative functions can be executed once for an entire group or "domain" of servers, making it unnecessary to manage all aspects of the network environment on a per-server basis.

- **The operating system simplifies the naming of resources for users, administrators, and programmers**. All applications can use the same naming conventions, defined by the operating system, for users, groups, distribution lists, domains, printers, workstations, and servers. Users and administrators benefit directly from this simplification.

- **Applications can fully integrate with operating system security services**. To control access to their services and resources, applications can take advantage of Windows NT security services. This frees the user from having to logon more than once, and frees administrators from adding users to multiple application-specific security databases. The programmer, in turn, does not need to create a custom database of users and perform authentication and access validation on his own.

- **Accessing low-level network protocols in programs is now much easier**. Much of the complexity of communications protocols is hidden from sight: users don't need to know whether they're running on TCP/IP, IPX/SPX or some other protocol. To access the network, applications and services simply rely on well-defined system services such as Distributed COM (DCOM) or Remote Procedure Calls (RPC), without concern for protocol specifics.

Roadblocks to Distributed Computing

Windows has made tremendous progress toward becoming a great platform for distributed computing. However, computing on a network of PCs is not yet where it could be: as simple in concept as computing on a single large machine, but more economical in its execution. Distributed computing promises to make servers work together seamlessly as a unit, yielding better fault tolerance and easier, less expensive scalability than can be achieved with a physically centralized computing environment. In short, distributed computing promises the best of both the centralized and decentralized computing worlds. It promises to surpass the benefits of the client/server model while drastically reducing its costs.

Establishing the ideal distributed computing environment requires completely integrating key networking services into the operating system. This will solve a number of difficult problems that users and administrators currently face in dealing with networked PCs:

- **Finding information is too hard**. As information becomes more dispersed, and as the control and management of information and resources becomes more decentralized, users are having a harder time finding what they're looking for. People spend too much time trying to understand and navigate the physical structure of the network, which has little to do with the information they are trying to find. What's more, they must sift through an overwhelming volume of information to find the exact pieces they're seeking.

- **There are too many directories and security systems to manage**. Because the operating systems do not yet offer server applications a generalized, shared "schema,"[1] many applications invent their own directory-like services or create application-specific add-on directories outside the one provided by the system. This leads to management headaches. For example, the administrator bears the burden of a system in which each application (data base, file system, mail system, and security system) handles replication of information independently.

- **The cost of managing a large distributed network grows at least as fast as the network**. The task of managing an ever-growing number of clients and servers is becoming more complex and costly. It's unrealistic to expect that the same model used for managing smaller networks that use one or two servers will work for large corporate networks or for a massive, public network like the Internet. The way networks of workstations and servers are managed must fundamentally change.

[1] An object "schema" is a definition of its properties and behavior. For example, the schema for a printer gives its description, location, name, driver, print queue, and the operations that can be performed on it, such as pause or resume.

- **Servers and networks of servers are still not robust enough**. Individual servers may behave like miniature mainframes, but they are more susceptible to certain kinds of failure. Networked servers can and must achieve a very high level of availability at a significantly lower cost than mainframes.

- **Large-scale public and enterprise networks need more flexible and comprehensive security options**. Intranet, Extranet (between businesses), and Internet network scenarios each have different security requirements. Security in a very large company needs to scale well to a large number of "domains." While many intranets are not as restrictive internally as public networks, they do need protection from outside access and they require strong resistance to attacks. Between businesses and on the Internet, security needs to be very restrictive and it needs to scale well to a large number of users.

- **Incremental scaling is still too hard**. When a server becomes overloaded, it is not always obvious how to apply additional servers to share the load effectively. Moving specific functions (e.g. the database on a Web server) to another machine will not divide the processing load optimally or evenly. Instead of placing the responsibility of load balancing on administrators, the ideal distributed environment should "auto-magically" readjust computing load when a server is added to the network.

Solving these problems requires a rock-solid foundation for building and running distributed services and applications. The Windows NT operating system is that foundation. Windows NT 5.0 includes a number of enhanced features that will yield immediate benefits for end users and administrators:

- Active Directory
- Distributed Security
- Distributed File System
- Management Console
- Active Server Pages (included in Internet Information Server)

Complementary technologies include the following:
- Clustering
- Message Queuing
- Component and Transaction Services

Making it Easier to Use and Manage the Network

In a distributed environment, Windows NT 5.0 will be more reliable, more scalable, and substantially easier to deploy, manage, and use than any distributed system created to date.

- **The Active Directory in Windows NT 5.0 makes it easy to find information**. Administrators, users, and applications can find information about people, printers, files, and other shared resources in a single place—the Active Directory—even if the resources reside on many different physical servers. Objects are assigned to logical workgroups or other organizational units rather than to individual servers. Therefore, users won't have to change how they find and name objects like files when administrators move them to different physical servers. This "location independence" is fundamental to making distributed computing simple to use and easy to manage.

  As with other system services, Windows NT 5.0 defines standard interfaces to the Active Directory (referred to as ADSI) so that other directories can integrate with it. In fact, the Active Directory can present information from multiple sources as a single system directory object. Some of those sources can even be dynamic—for example, the printer status or the contents of the print queue.

  The Windows NT native protocol for directory access is the industry standard LDAP, which allows for extensive interoperability with directory services from other vendors.

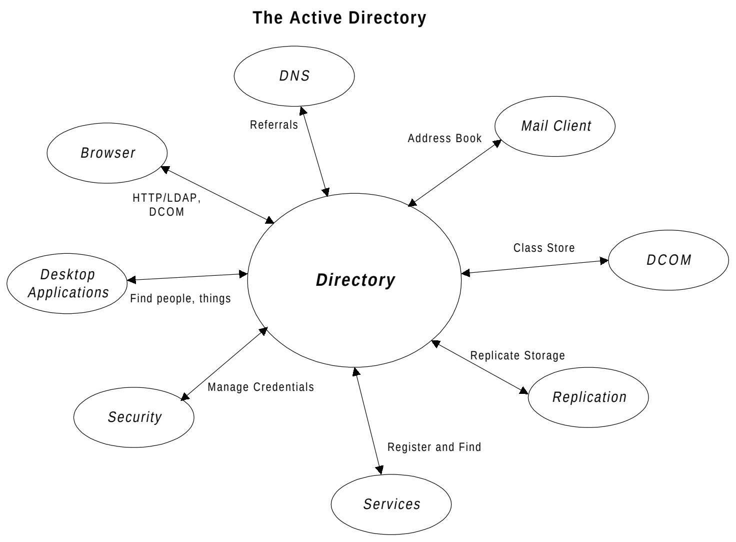
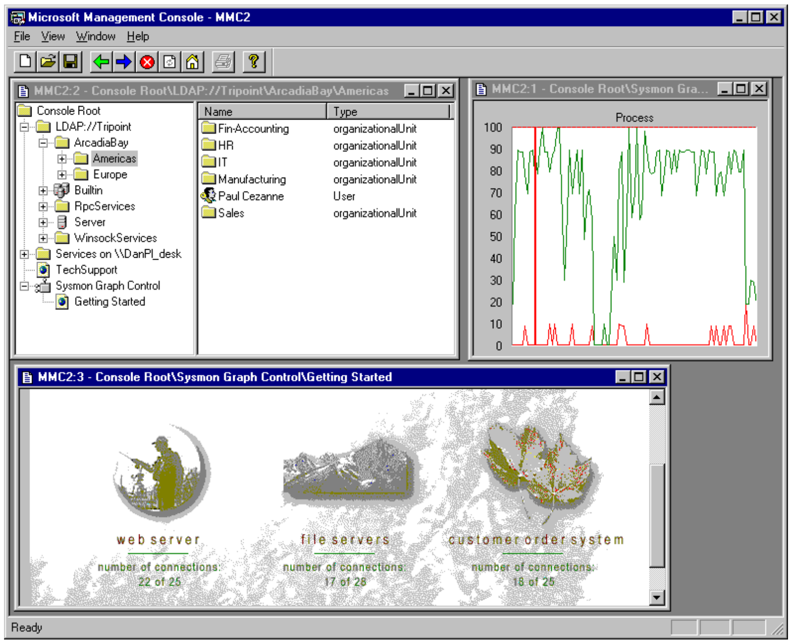


Figure 2. The Active Directory provides a logically centralized location for finding shared resources. This insulates users and administrators from having to navigate the physical structure of the network. Instead, the Active Directory gives them a unified view of the network that doesn't change as the physical network structure evolves.

- **The Active Directory removes the need for numerous application-specific directories, making it easier to manage the network**. Even the most demanding applications and services can take advantage of the Active Directory because it is extensible. Applications can add or change information in existing object "schema" or add new object classes. As applications and services evolve to make use of the directory, it will naturally become the backbone for network management, especially in large enterprises.

  The Active Directory combines the best features of Domain Name Services (DNS) and X.500. For example, since it uses DNS as the global backbone namespace, it uses DNS to look up Lightweight Directory Access Protocol (LDAP) services. It also integrates DNS with directory storage and replication. Though the Active Directory uses the X.500 data model, the implementation is more light weight – the Active Directory uses the LDAP protocol instead of DAP, and it uses a combination of Kerberos and public key security that is tightly integrated with Windows NT and BackOffice services, including file and print.

- **The Microsoft Management Console greatly simplifies management of network services and applications**. Administrators will be able to manage all services, applications, users, and resources in an enterprise network through the Microsoft Management Console (MMC), a comprehensive tool that ships with Windows NT 5.0. MMC operates over the Active Directory. Applications and services can expose information and operations to the Windows NT management framework as directory objects. Special-purpose management tools can integrate with the Microsoft Management Console with extensions called "snap-ins."



Name	Type
Fin-Accounting	organizationalUnit
HR	organizationalUnit
IT	organizationalUnit
Manufacturing	organizationalUnit
Paul Cezanne	User
Sales	organizationalUnit

Figure 3. The Microsoft Management Console allows administrators to manage all system services and applications from a single, customizable, and comprehensive tool.

The management console serves as the host for scripts written in any language that can operate over any combination of objects in the directory, regardless of what vendor, service, or application created those objects, and regardless of where those objects physically reside. In addition, administrators can create, save, and exchange any number of console configurations, which enables very concrete delegation of responsibilities and tasks. The goal is to let administrators create a single, customized view of all management activities.

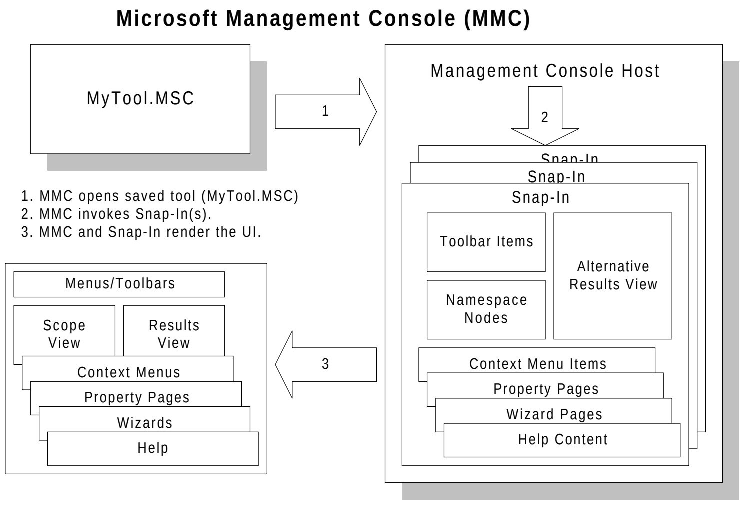


Figure 4. The Microsoft Management Console snap-ins allow the administrator to create a single, custom view for all management activities.

- **The Zero Administration Initiative for Windows (ZAW) allows for completely centralized management of all workstations in an enterprise**. Instead of forcing the administrator to manage all enterprise workstations remotely, one at a time, and in real time, Windows NT 5.0 and “Memphis” take a different approach. All configuration and state information for users, applications, and machines is centralized so that administrators can more easily access and manipulate it. Once user and machine group profiles are established, administrators don't need to upgrade the operating system manually or install applications onto individual machines. Instead, a policy dictates which users require which application software, and which computers require which operating system software. The system then updates application and system software automatically—and keeps it up to date, even if policy evolves.

Storing all workstation configuration and state information on a central server means that users can effectively “roam” from one physical workstation to the next. When they logon to any machine with the same user name and password, they see the same desktop configuration, set of applications, and group of files. Administrators can also replace a user's client machine without backup and restore operations—the system simply regenerates the correct software installations based on the user's profile and the requirements of the hardware.

The NetPC, a joint effort between Microsoft, Intel, and leading hardware vendors, is specifically designed to enable centralized, controlled IT management. It is a fully functional Windows-based PC that boasts a uniform hardware configuration and a sealed state that prevents user-access to serviceable parts. When deployed in combination with ZAW features, the NetPC provides a simplified computer working environment, compatible with existing applications and PCs, that greatly reduces management costs.

- **New system and network services will greatly increase resource availability and system robustness**. All Windows NT 5.0 services are designed to exploit distributed computing to enhance the system's resistance to failure. No single failure in the network will cause Windows NT's core distributed services to fail.

  Clustering for Windows NT, which debuts in mid-1997, provides fail-over support for paired servers.[2] Should one server fail or require downtime for routine maintenance, the other takes over. When all is well, the clustered machines are free to process network requests independently, improving performance by load-balancing requests for server resources between the physical machines.

- **The security protocols and services in Windows NT are more safe, flexible, and scalable than ever**. The Windows NT 5.0 updated security system offers stronger resistance to attacks. State-of-the-art public (RSA) and secret (Kerberos-based) key technologies provide simple, integrated, and comprehensive security services. Private security allows for secure management of network resources and gives applications easy access to information stored in the Active Directory. Public key is particularly important for very large enterprises, public networks, and inter-enterprise security. Although they address different security needs, public and private security share a single infrastructure within Windows NT 5.0’s distributed security services. The security services easily scale from a small local area network, to a large corporate intranet, to a “virtual private network” across the Internet (see Figure 6), to the Internet at large.

[2] Later releases will allow for additional servers in a cluster and will add advanced features for automatic load balancing across machines in a cluster.

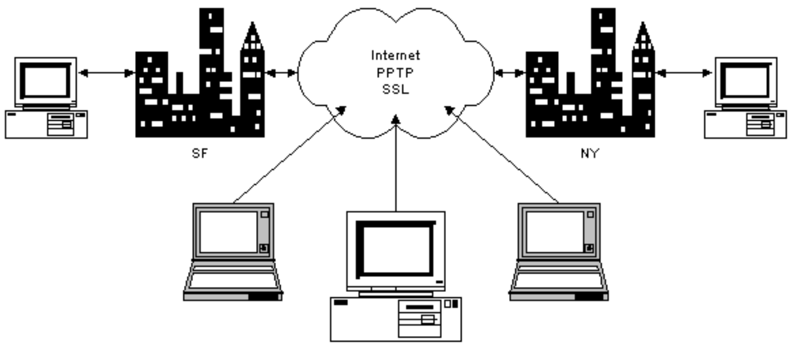


Figure 5. Using the Point-to-Point Tunneling Protocol (PPTP), businesses can connect private networks in different geographic locations across the public Internet instead of via expensive leased lines. Remote users can also access the corporate network through the Internet.

Windows NT 5.0's security services incorporate X509v3 certificates, which are used during strong authentication to identify users trying to gain access to the directory. Security for X509v3 certificate holders is integrated with access checking for Windows NT 5.0. The system maps an identifying property of each certificate to a property on a principal (an object used for security checking). Thus, when the system determines access rights for the principal (e.g., during an ACL check), it can use that principal to determine access rights to the directory.

It is now possible for a component running on a server, such as a Web server, to impersonate user identities when accessing other servers, such as a SQL database, on their behalf. And Windows NT 5.0 makes it possible for any application to use the system's public key services, for example to encrypt an object so that only a specific recipient can read it, or to "sign" an object or message so that the recipient can securely verify its origin and integrity.

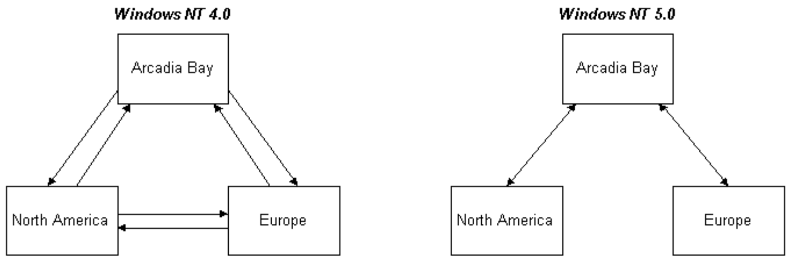


Figure 6. In Windows NT 4.0, trust relationships between "Arcadia Bay" and the domains "North America" and "Europe" must be established in all directions. In Windows NT 5.0, trust relationships can be transitive through a parent domain (in this case "ArcadiaBay"), reducing the number of relationships the administrator needs to maintain.

The addition of transitive trust relationships between individual domains makes a huge difference in the ability of Windows NT to scale to larger enterprises. Smaller "domains" don't need explicit trust relationships with all other domains in the larger organization. Instead, they can inherit trust relationships established by a parental domain at the root of the organization, thus greatly simplifying ongoing management of directory and security services.

With Windows NT 5.0, Microsoft is building a model for full-scale Internet and intranet computing that is much simpler to use and manage than anything out there today. Within corporations and on the Internet, Information Systems groups will be able to scale their service offerings, increase robustness, and ensure availability using "building blocks" of servers without worrying about how to manage individual machines. And while centralized planning can yield benefits down the road, it is not required; Windows NT 5.0 allows for the real world "grass roots" growth of workgroups that can be merged into an enterprise network at a later time.

The Challenges of Programming in a Distributed Environment

These improvements are important steps toward making Windows NT a truly distributed platform, however there is more to do. Applications and services added to the system must preserve the simplicity of the model we're striving to create for the end user and the administrator. This requires radically simplifying the task of creating such programs. We need to transfer the simplicity of programming in a single machine environment to the world of distributed computing without actually retreating to a mainframe model.

In Windows NT, the same foundation services that make applications and services easier to manage will also make them easier to write. The result will be a new generation of distributed components that are significantly more robust, and that scale better to extremely large networks. Technologies such as Internet Information Server and Microsoft Transaction Services build on Windows NT's foundation services to make the creation and deployment of distributed components fundamentally easier and the results much better than anything developers or customers have yet experienced.

To accomplish true distributed computing on the scale of the Internet, server applications must not only fit into the logically centralized, physically decentralized view of the network, they must *promote* it. Conquering truly scalable, robust computing—the kind reliable enough to run a business—will require that distributed components fully integrate with the operating system's distributed services. When components let the system manage how they are distributed and activated in concert with system services, they will be easy to deploy, they will scale simply and seamlessly, and they will achieve the same high level of availability as the distributed system itself.

Up to this point, however, it has simply been too difficult for software developers to "do the right thing" for end users and administrators in a distributed environment:

- **Current communication methods and messaging models are inadequate for large public networks**. Synchronous communication through Remote Procedure Calls (RPC) requires that clients wait for a response before they can continue executing code. Asynchronous messaging alleviates this type of congestion, but it is very difficult to program. Both styles of communication rely on a continuous live network connection, which is hard to maintain in many network environments, like the Internet. Components should not have to deal with these subtleties or specific details about HTTP, TCP, or RPC in order to communicate in a reliable way.

- **When an application or service operates in a distributed environment, the possibilities for failure increase dramatically**. As a result, developers end up writing a lot of client and server code to deal with a staggering number of error state combinations. Monolithic applications fail as a unit, but when a component fails, it needs to recover gracefully so that applications relying on it can keep running. Treating groups of operations as a single atomic unit is still necessary, but in a distributed environment it is hard to perform operations this way, particularly when an application scans more than one storage system for pertinent information.

- **Coordinating and scaling the components of a distributed application is difficult**. It requires a lot of overhead and added complexity to determine the following:

    - How components should interact with components from other applications
    - Which communication transports to use (HTTP, TCP/IP, IPX/SPX)
    - Where individual components should run-on which server, in which process, or in which thread
    - How an application recovers if one of its component fails

- **Writing server applications requires intricate knowledge of how low-level system services work**. For example, to achieve efficiency and robustness, developers must write complex code to handle processes, threading, transactions, security, and caching—for each and every server application they build. This is a tremendous time sink that can take up to 30 to 40 percent of development time. [3]

- **Operating systems do not provide a comprehensive set of interfaces for accessing key distributed services or "legacy" data and systems**. While modern operating systems provide exhaustive support for multi-processing, multi-threading, file services, graphics, and low level network access, they do not provide complete, simple-to-use interfaces to directory, and distributed security. Data access services are also critical system services. Customers who have extensive investments in legacy database systems want to access them from new technologies without having to upgrade or re-architect the legacy systems themselves.

- **Creating dynamic Web content is too cumbersome**. The most commonly used technology for building dynamic Web content, the Common Gateway Interface (CGI), embeds text, layout, and graphics, and other web page elements in CGI script. As a result, web page authors must sift through source code to make even small changes that then require recompilation. The CGI mechanism forces HTML designers to be programming literate, and it forces programmers to understand HTML design issues.

The Internet's Impact on Distributed Computing

Making it easy to create distributed components is increasingly important since the Internet has pushed more logic onto servers. In the traditional client-server model, the client ran a significant portion of the application-specific code, but in the Web model, it focuses almost exclusively on presenting information. In many cases, the Web server now runs nearly all business or application logic, which shifts significant computing load to the server. As a result, code written for the server must scale better than ever before. To reach the Internet, more and more programmers who now focus on the single user desktop are facing the daunting task of writing server code effectively.

Furthermore, companies want to build one common infrastructure that encompasses both their intranets and the Internet. This is difficult to do without a well-established programming model for the server. A skeletal Web server programming model does not address scalability, distribution, security, and many other issues. For example, stateless Web servers require applications to reinvent state with clumsy mechanisms like "cookies." In fact, today’s Web server is in many ways a step backwards from traditional client server computing and the kinds of services available through the RPC server runtime.

It is imperative that server programming become more accessible to average developers, such as the more than 2 million programmers who use Visual Basic®, and that what they write scales effectively.

In summary, programming in a distributed environment must:

- Give end users and administrators the illusion of a "single server"
- Be easy, requiring little or no advanced training or retraining
- Have a single, unified programming model for the Internet and intranet
- Allow for tight integration with comprehensive operating system services
- Produce robust, scalable application components that are easily distributed and managed

Making it Easier to Program in a Distributed Environment

[3] Source: Mitchell I Kramer, Patricia Seybold Group, Nov. 1996.

Microsoft's server technologies build on the distributed features of Windows NT 5.0 to address these and other fundamental issues. The Microsoft Transaction Server, for example, offers a comprehensive runtime environment, accessible to all programmers, that manages distributed components. The more developers write simple components that rely on system services to manage state, resource sharing, and error recovery, the more systems can easily manage them and achieve scalability. Under certain circumstances, however, system-level programmers will continue to write applications and services that are not fully component-based and don't take advantage of the runtime environment. This flexibility is key to the success of the platform: it allows the developer to operate at the level in the system appropriate to the problem.

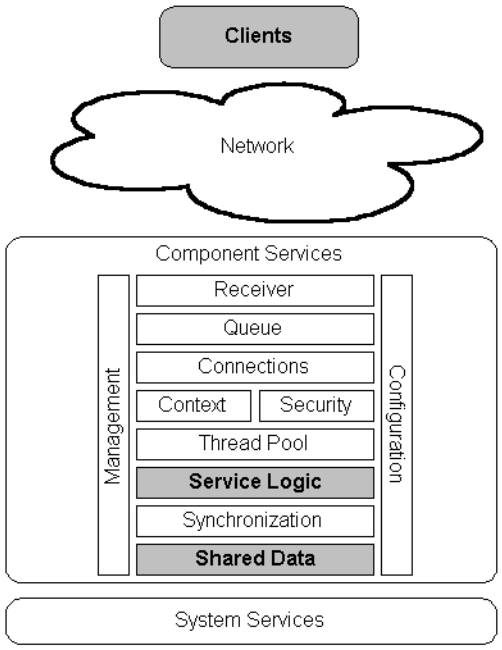


Figure 7. Microsoft Transaction Server makes it easier to create server applications by managing low-level system services, coordinating distributed components, and hiding complexity from the programmer.

With Windows NT 5.0 as the foundation, Microsoft offers a comprehensive server platform with the following benefits for programmers:

- **Microsoft's server technologies insulate developers from server programming complexity**. They let developers focus on their real task—implementing business or application logic—by doing the following:

    - Providing ready-made "server plumbing" that handles low-level tasks like managing processes and threads or creating, activating, and destroying object instances.
    - Treating all component pieces of an application as a single package, determining how to dynamically install, configure, and run them.
    - Managing communication between components using DCOM. It handles connection management, authentication and access validation, and context management.
    - Managing resources, performing concurrency control and lock management over shared resources, hiding the complexity of preventing race conditions and starvation, and breaking deadlock conditions.

- **Microsoft's server technologies add flexibility to managing server applications**. They make it possible for a component to run virtually anywhere on the network. The program that calls the component doesn't need to know where it is physically running. System administrators (or the system itself on the fly) can therefore decide on which physical machine a component will run based on the current state of the network. Developers do not hard-code these decisions into the components themselves.

- **Microsoft Transaction Server simplifies error handling through transactions**. It brings the best ideas from transaction processing monitors to the world of distributed components. An application is unaware that an operation it requests may actually span multiple machines or components; it simply begins a transaction, makes a series of calls (that may execute components all over the network), and requests a commit. Unless the whole series of operations complete successfully, Transaction Server will roll all of them back. When combined with clustering, Transaction Server's management of distributed components makes it possible to achieve extremely high availability.

- **Windows NT, along with technologies like message queuing, provides a complete communications infrastructure**. It guarantees interoperability between components and manages lower-level communications so that programmers do not have to choose specific transports for different network conditions.

  - Because they build on Windows NT, Microsoft's server technologies inherit communications mechanisms from the operating system: TCP/IP, RPC, RAS, PPTP, etc.
  - COM ensures interoperability between components with a consistent architecture and standardizes the basic mechanisms components use to communicate with each other. It also abstracts various messaging styles, including synchronous and asynchronous transport services.
  - "Falcon" provides message queuing and asynchronous communications. Two computers that are communicating with one another don't have to be up and running at the same time. The system makes sure messages are received. Components transparently benefit from this functionality – no special programming is required.

- **Components can easily integrate into the system's Web services**. Active Server Pages, a feature of the Internet Information Server, combines HTML, server-executed scripts and components to generate web pages dynamically. Active Server Pages act as the "presentation services" liaison between a server component and HTML or HTTP clients like browsers. Scripts can invoke components through standard OLE Automation, enabling access to databases, mainframes, or any service running on the server. Active Server Pages are much simpler to create and modify than CGI scripts, clearly separating components from HTML design issues. Web designers create pages in standard HTML syntax that is compiled Just in Time (JIT).

- **Server components can transparently access legacy data**. Through "Cedar" technology, components running on the Active Server can interact and launch "work with and invoke" applications running on a mainframe (to access legacy data). For example, an application running on Windows NT Server can invoke transactions on legacy CICS/IMS mainframes over LU 6.2 connections and return the results in HTML so they can be displayed on a Web browser.

- **Developers can build server components using familiar tools**. Because COM components written in any programming language interoperate naturally and seamlessly, developers can use a wide range of tools from Microsoft, Borland, Powersoft, Sybase, Symantec, ORACLE, IBM, Micro Focus, and others to build server components. It's now possible for the average programmer to accomplish with high-level languages what once required low-level tools and guru-level programming expertise.

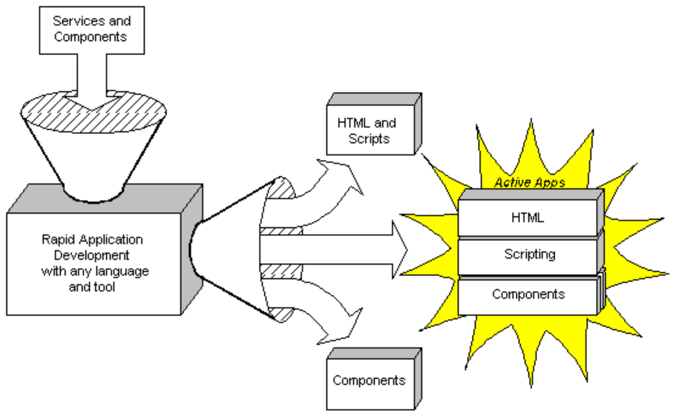


Figure 8. Existing tools that support COM can access system services and reuse off-the-shelf components to create new components, applications and dynamic web pages.

What the Future Holds

In the future, system services such as the file system and the registry will become transaction-based, making the set of services transacted components can call much broader. For example, applications will be able to ask the system to roll-back complex, aborted operations such as the unsuccessful installation of an application. New and existing system services will be made available through COM interfaces, making them even more accessible from high-level languages and tools.

Windows and Windows NT will continue to evolve, strengthening the distributed platform with features that improve simplicity, take advantage of hardware advances, and reduce the total cost to corporations of owning and maintaining networks of computers. Features that are planned for the timeframe beyond Windows NT 5.0 include the following:

- Multi-node clustering and load balancing
- Multi-user Windows NT, integrated with clustering for scale
- More sophisticated integration of the Windows shell and the Active Directory

For more information

For more comprehensive, detailed information, check out the Windows NT Server Forum on the Microsoft Network (GO WORD: MSNTS). To access information via the World Wide Web, go to http://www.microsoft.com and select Microsoft Sitebuilder, intranet, MSDN, NTServer, or BackOffice.