Chapter 5 Window Systems on Sun Workstations

Sun is making fundamental changes to its window system architecture this year, as well as announcing new window system products associated with the AT&T UNIX standardization effort. Sun is phasing in a new industry standard window system with toolkits based on the OPEN LOOK™ user interface and the X11/NeWS window server as it phases out the current proprietary SunView™/SunWindows™ system. These changes will not only make it easier for end-users to learn new applications, but also it will provide a more portable development environment for programmers.

Sun has been shipping window systems from its inception and has strong current offerings. In addition, Sun has a solid transition strategy such that products developed on the current window system can be easily moved to the new window system. Until the new window system is available, developers are encouraged to use the current system, both for current products and as a development platform for the new system.

To better understand the ramifications of the window system changes, it is necessary to examine Sun's offerings in terms of the general structure of window systems. So this chapter explains window systems and describes Sun's present and future window system offerings. Finally, it discusses the transition to the new window system for present customers.

5.1 Window System Architecture Primer

Window system architecture can be separated conceptually into five layers (See Figure 5-1.)

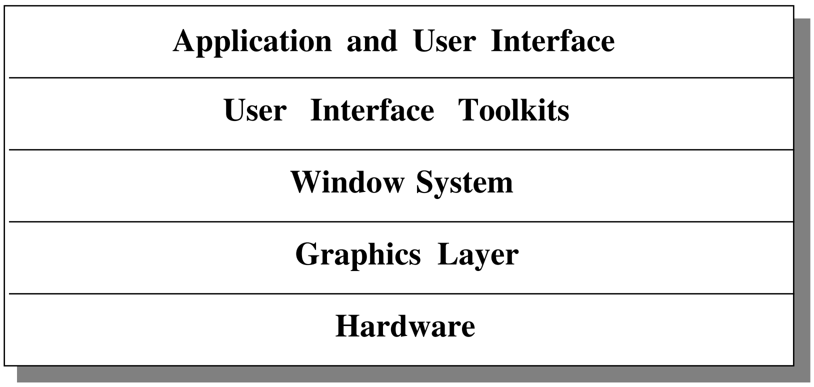

Figure 5-1. Window Systems Architecture

The foundation layer is the hardware at the bottom of the layering scheme.

The second and third layers, respectively, contain the graphics layer that the window system sits on and the window system itself. The graphics layer controls the manner in which the window system accesses the display. A window system based on the pixel model draws to the display by pixel, or picture element, whereas one based on a stencil/paint model is device- and resolution-independent, creates images based on a world coordinate system, and maps the image to a display device without regard to pixels. The window system layer is the software platform that provides a system interface and manages the creation of windows, receiving events from devices and providing notification for selections made in a window.

The fourth and fifth layers implement and define the user and programming interfaces. The user interface toolkit layer, the fourth layer, provides an applications programming interface (API) to the window system layer; it implements the user interface definition (look and feel) in the topmost layer.

A toolkit is a high-level programming interface that provides developers with pre-cut user interface building blocks. The toolkit not only makes application development easier but ensures that the user interface remains consistent so developers use the same building blocks. Toolkits help developers create user interface prototypes quickly and easily and help establish uniformity of user interface across applications while providing a simple and easy-to-use programming interface. The user interface layer conceptually provides the look and feel to the user, which makes windows easier to use and provides a consistent method of manipulating windows and applications within them.

Sun's current products; SunView and NeWS, and future products; the X11/NeWS windowing system, the View2 toolkit, the NDE toolkit, the two types of graphics layers, and the OPEN LOOK user interface specification, can be mapped to this layered construct. (See Figures 5-2 and 5-3.)

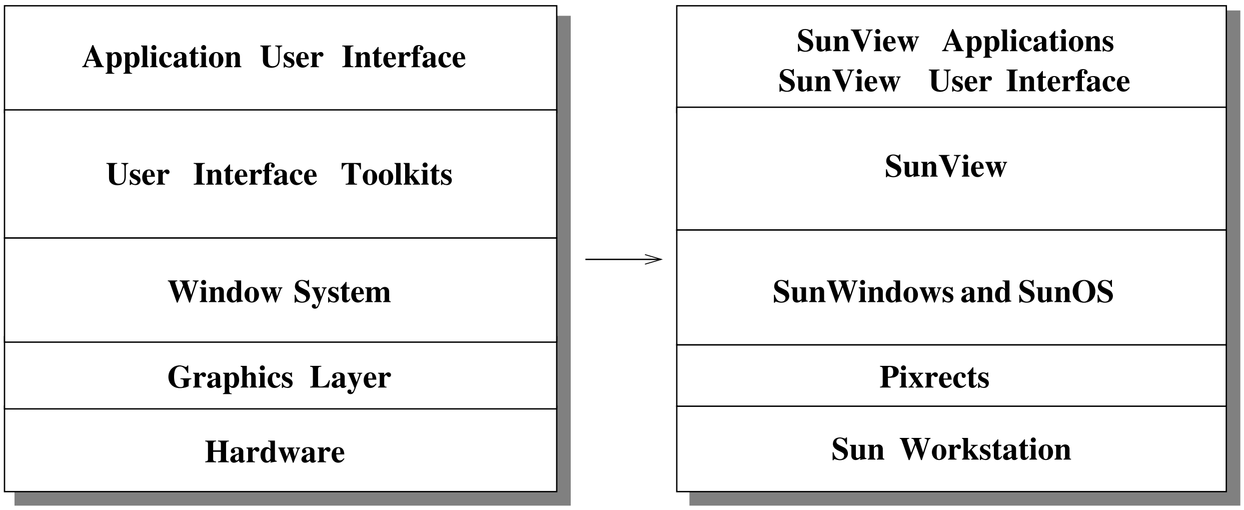

Figure 5-2. Current Window Architecture

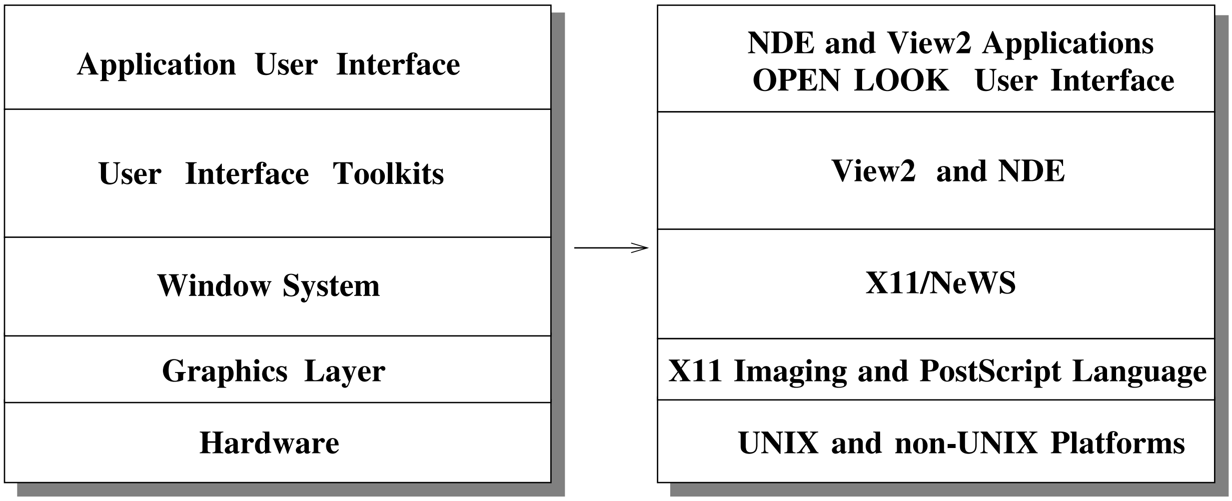

Figure 5-3. Future Window Architecture

5.2 User Interfaces

It is important to note that a user interface specifies a "look and feel" for a user application. A user interface is not a software toolkit and does not affect an API. Sun's future toolkit offerings will implement the OPEN LOOK specification across multiple toolkits.

5.2.1 SunView User Interface

The SunView user interface is Sun's current proprietary user interface. There is no specification of the SunView user interface; it is defined as anything that can be produced with the SunView toolkit. The toolkit is powerful and flexible. However, the side effect is that one application built with it may have a vastly different look and feel from another.

5.2.2 OPEN LOOK User Interface

The OPEN LOOK user interface is simple, consistent, and efficient

The OPEN LOOK user interface is a non-proprietary graphical user interface designed by Sun and proposed as a standard for bit-mapped displays. The OPEN LOOK user interface is not a toolkit; it is a specification for a look and feel. The user interface works on many different hardware platforms and operating systems, both with today's technology and that anticipated for the future. Applications that conform to the OPEN LOOK specification share the same look and feel and are easier for customers to learn and use than programs developed with an *ad hoc* user interface. (See Figure 5-4)

The three goals of the OPEN LOOK user interface are simplicity, consistency, and efficiency. Simplicity means that OPEN LOOK applications are easy to learn.

Consistency means that users who know one OPEN LOOK application can quickly learn another. Efficiency means that users can use shortcuts to speed user interface operations.

In developing the OPEN LOOK user interface, Sun made sure that it worked well in a multitasking environment, at a variety of screen resolutions, and for a wide variety of applications. Wherever possible, OPEN LOOK works in the way users familiar with other user interfaces would expect. For more information, see the *OPEN LOOK Functional Specification* and the *OPEN LOOK Style Guide*.

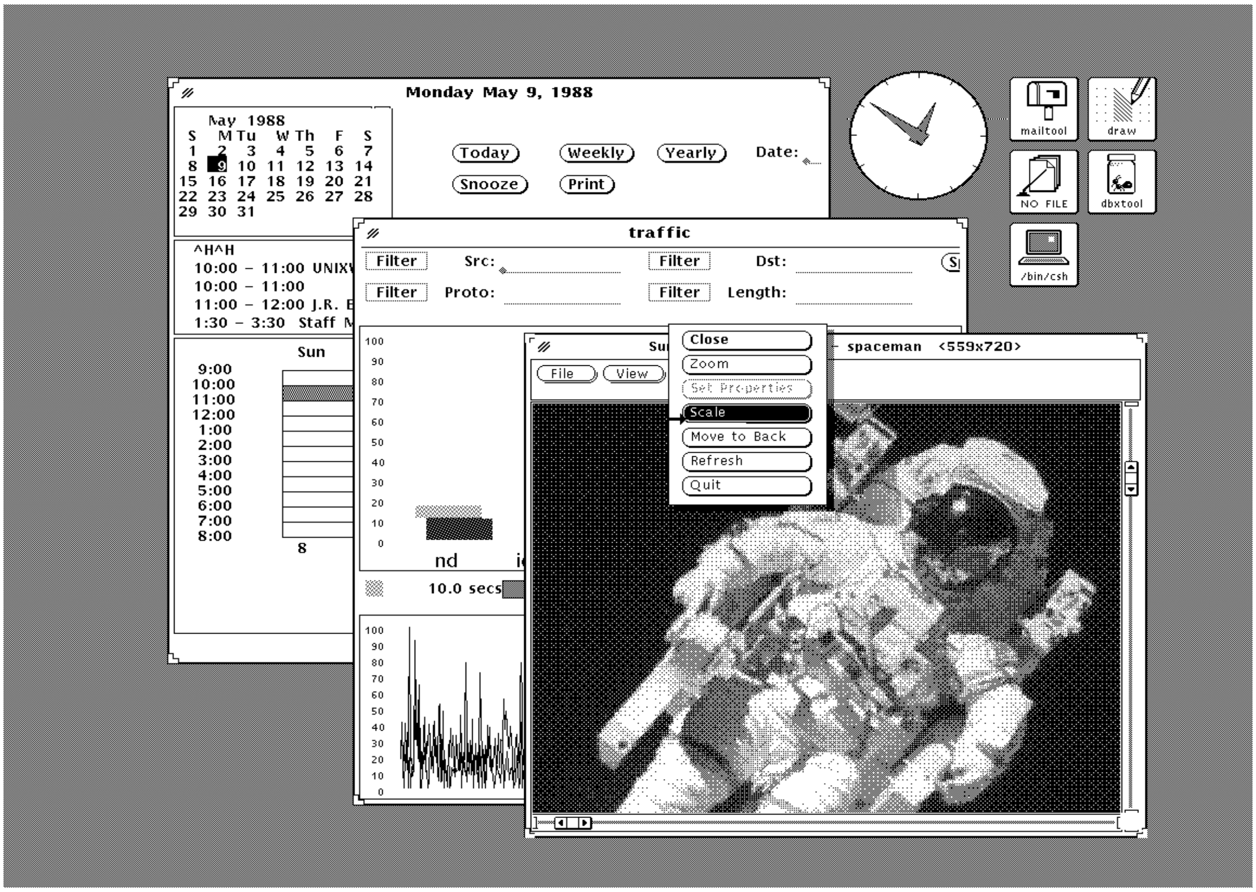

Figure 5-4. The OPEN LOOK User Interface

5.3 Toolkits

Sun's current user interface toolkit is SunView. Future offerings will include View2 and NDE™, both of which implement the OPEN LOOK user interface. SunView and View2 have a similar API. Thus, programs developed with SunView are easy to convert to View2. NDE, the NeWS Development Environment, is both a toolkit and an advanced development environment designed to remain state-of-the-art for many years. NeWS programs may also be developed using a small class-based toolkit that is shipped with NeWS.

5.3.1 SunView Toolkit

SunView is Sun's most popular and mature window toolkit

SunView is Sun's current toolkit for building window-based applications. SunView has been a popular and stable environment on Sun systems for the last three years, and most third-party applications are based on SunView. Like most existing user interface toolkits, SunView is kernel-based and applications built for SunView can only run locally.

SunView provides a rich set of components for building applications. These include drawing surfaces (called canvases), editable text windows, and TTY emulation packages as well as a full assortment of user interface components including alerts, buttons, icons, menus, control panels, sliders, and scroll bars.

SunView applications are built in a top-down fashion. First, a window is built, then user interface components and drawing surfaces are added. Rather than having the application poll for events, SunView is notification based. The application registers event handling routines with each SunView component used. Then, when an event occurs within that component, the applications event handler is called. What constitutes an event depends on the component the event occurs. For example, a button selection reports a mouse down, while a menu selection reports the selected menu item. Applications can be informed of other types of events such as signals or the arrival of new data to a pipe, socket, or other UNIX file.

Some applications, such as those from a TTY-based environment, are difficult to convert to the event notification structure. In such cases, applications can circumvent the normal event notification procedure and directly receive the keyboard, mouse, and window events.

Graphics are drawn on a SunView canvas with calls from the Pixwin library. All SunView graphical elements, including canvases, have an associated Pixwin, which is the primary object used by the Pixwin library. Pixwins are, in turn, based on Pixrects. The direct use of Pixrects is discouraged in most cases due to portability issues. Pixwins provide basic raster functionality including lines, polygons, rasterops, and text. Pixwin does not support high-level graphics operations such as shading, rotated text, and transformations.

The performance of Pixwin-intensive procedures can be increased in a number of ways. Whenever possible, use the polyline and polypoint routines instead of the line and put calls. Careful use of the locking and batching capabilities improves performance. Pixwin performance is also increased by using the graphics accelerators described in Chapter 6. Pixwins (and Pixrects) are also discussed in that chapter.

The SunView1 Programmer's Guide describes the programmer's interface to SunView and how to write a SunView program.

5.3.2 View2 Toolkit

View2 provides the API of SunView with the OPEN LOOK User Interface

The View2 toolkit, the successor to the SunView toolkit provides the API of SunView with the OPEN LOOK User Interface. In its first release, it is a developer's toolkit and is based on a preliminary version of the OPEN LOOK user interface. For Sun and UNIX developers, View2 is the fastest path to implement applications conforming to the OPEN LOOK standard.

Because of its high degree of compatibility to the SunView API, View2 toolkit provides an easy transition for applications developed on SunView. Even though the transition includes a move from a kernel-based to a network-based window system, conversion of applications is easy because of the similarity in APIs. Since View2 is network-based, View2 applications can run across the network. The View2 toolkit will be available at the same time as X11/NeWS.

5.3.3 NDE Toolkit (NeWS Development Environment)

Sun's advanced development toolkit, NDE, will be available in 1989

The NeWS Development Environment toolkit (NDE toolkit) is an advanced toolkit for the development of NeWS-based applications. The NDE toolkit provides support for the development of OPEN LOOK Graphical User Interface compliant applications. In addition, the NDE toolkit provides a set of layered interfaces giving the developer the freedom to select the programming model that best suits the architectural needs of the application. Because, the NDE toolkit is a NeWS-based development environment, the NDE toolkit supports the development of networked-based applications and provides full access to the PostScript Language imaging model.

The NDE toolkit is comprised of two main components: a client component and a server component (See Figure 5-5.) As stated above, each main component presents the application developer with a set of interfaces to choose from.

The highest level of the client component provides an object-oriented development environment. The object-oriented interface allows the application developer to program in C++, or standard C, using a preprocessor to support the object model. At lower levels of the client component, the application developer is presented with a more traditional "call back" of services. The application developer may choose from C, Fortran, or Pascal for programming at these levels.

The server component provides an object-oriented PostScript Language interface. The various layers allow additional user interface components to be developed. Additions to the server component can be made available to an application developer who is using the client component.

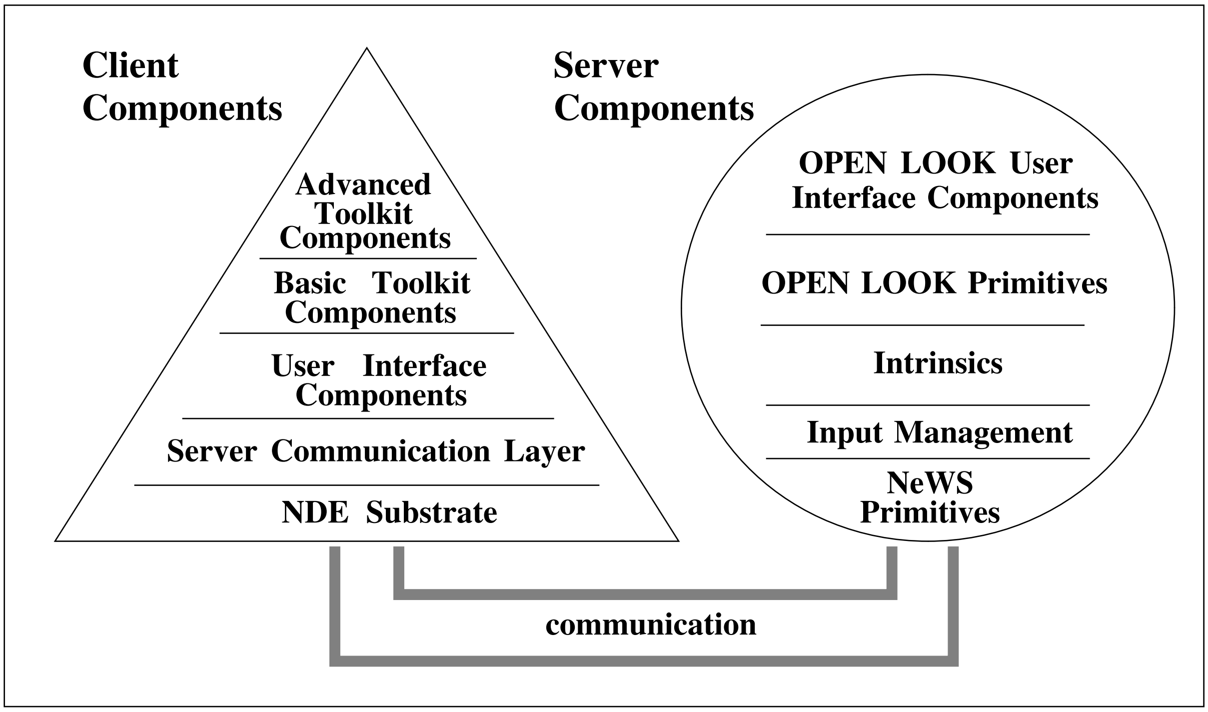

Figure 5-5. NDE Toolkit Layers

By providing a basic set of objects and the mechanisms to extend them, the NDE toolkit provides an extensible development environment that allows application developers to build on previous efforts. The combination of powerful object-oriented development environment and PostScript Language imaging model will provide a platform that is able to adapt to new technologies in the years ahead.

5.3.4 NeWS Lite Toolkit and CPS

NeWS is a window platform, not a user interface toolkit. It does not provide the high-level packages that SunView and View2 supply, such as menus, windows, and scrollbars. NeWS does have a small package that helps developers build these higher-level packages: the NeWS Lite toolkit. NeWS Lite has an object-oriented structure and a Smalltalk-like class mechanism.

The Lite toolkit can be used to build window programs entirely in the NeWS server environment. However, to take advantage of the networking capabilities of NeWS, an application must be partitioned into computational (client) and window (server) sections. With the built-in dynamic extensibility of NeWS, it is feasible to put most if not all of the user-interface in the server itself, thus improving the performance and responsiveness of the user interface. Highly computational parts of the application can be written in C and interact with the server user interface packages in the server using the C to PostScript Language (CPS) package. Using CPS, applications may be partitioned in a customized fashion. Although the CPS interface is the most commonly used, any language that can access sockets can be used to build such a partitioned application.

5.3.5 X11 Toolkits

Like NeWS, X11 is a window platform that does not impose a particular user interface. X11 also provides a class mechanism, X intrinsics, for building toolkits. Objects created with X intrinsics have a standard programmer interface and are known as *widgets.* It is possible to create a complete user interface toolkit by using X intrinsics and widgets.

X11 for Sun workstations is available from the MIT X Consortium. The MIT distribution also contains a toolkit, known as Xt, or the Athena Widgets, to ease the task of programming X. This toolkit is built on top of the X intrinsics package and supplies many of the popular user interface objects.

5.4 Base Window Systems

Sun's current window system is SunWindows. The new window system for years to come is the X11/NeWS merged server. Note that the server itself does not implement the OPEN LOOK user interface, but rather to the toolkits and applications.

5.4.1 SunWindows

Sun's current window products include SunView and the SunWindows window system, shipped as a standard part of Sun's operating system release for the past four and one-half years. Different parts of the SunView/SunWindows platform map to the layered window system architecture. (See Figure 5-2 and 5-3).

The name SunView actually refers to two separate layers of the window system architecture: the user interface toolkit and the user interface itself. SunView is the name of the current *user interface toolkit* used by developers to create application programs. SunView is also a *user interface* with an implicit definition and not a rigid specification. The SunView user interface toolkit implements the API that many customers have been using to develop applications for the SunWindows window system.

SunWindows, consists of the pixwins and pixrects library routines that are implemented on the pixel-based pixrects graphics layer, as well as the notifier, selection service, event mechanism, and screen-locking facilities. To enable applications written for SunView to continue to run on the mature SunView/SunWindows window platform, SunView will continue to be shipped with each operating system release for at least the next two years. However, in order to take advantage of the network-based window system X11/NeWS, and the OPEN LOOK user interface, we urge developers to convert their applications to conform to View2, X11/NeWS, and the OPEN LOOK user interface. (See "Product Transition" in this chapter.)

5.4.2 X11/NeWS

X11/NeWS is a server-based window system combining Version 11 of the X Window System from MIT, and the NeWS window system, which Sun developed and has been shipping since April 1987. X11/NeWS provides the advantages of both the X window system and the improved performance, stencil/paint imaging model, and the printer integration features available with NeWS. As a server-based window system, it is divided functionally into two parts: the *window server*, which provides the window management, protocol interpretation, and event queuing, and the *client*, or the application that uses the window server.

The server/client model is familiar for many NFS users, but less familiar for window system users. The window server is resident at the workstation that bears the display, "The server is where the screen is." The server interacts with client applications to share the display resources and the screen management capabilities of the window system. The client, or windows application, can run either on the same local workstation or can be at a remote machine with or without a display.

The client application communicates with the server via interprocess communication. In the case of the X11/NeWS window system, the interprocess communication is X11 protocol for an X11 client, NeWS protocol for a NeWS client. (See Figure 5-6.)

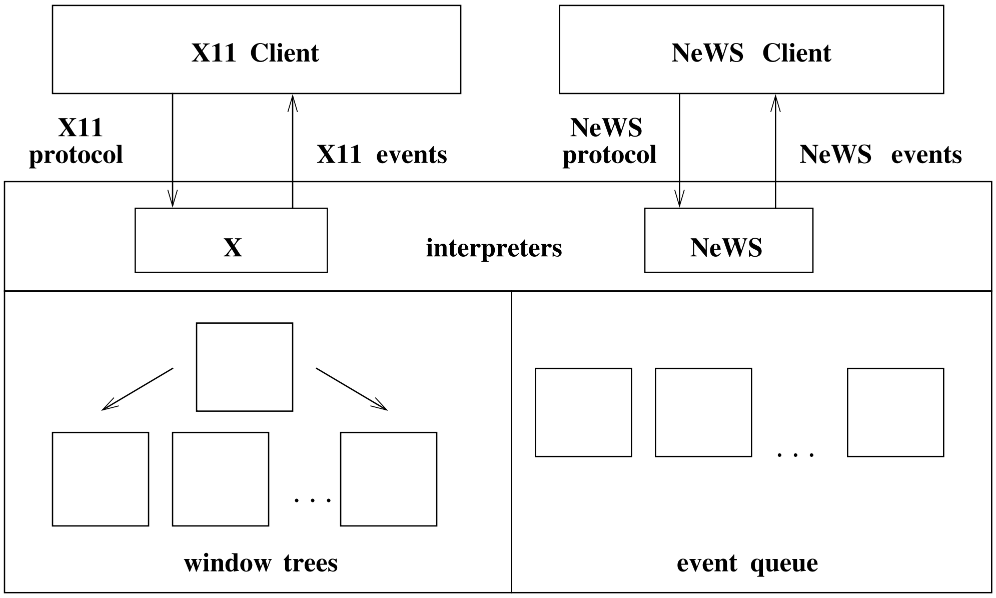

Figure 5-6. X11/NeWS Server Structure

Connections to the client applications are managed by two adjacent interpreters, sharing a common underlying graphics library. The server processes the protocol from the client, creates window events, and receives events from the keyboard and mouse, which return to the client as events to process. Hierarchical window structures (or window trees) are managed by a shared window manager. Both X11 and NeWS client applications can run simultaneously on the X11/NeWS server.

By providing the complete capabilities of both NeWS and X11 in the X11/NeWS merged window system, users can simultaneously run both types of applications. Although many applications developed on NeWS 1.1 or non-Sun X11 toolkits will run without change on X11/NeWS, some will require minor modifications. For a comparison of the two systems, see Figure 5-7.

The merged X11/NeWS window system will be included as a standard part of enhanced UNIX System V.4.

Window System Feature	**X11**	**NeWS**
language	C/Xlib (Raster Ops)	PostScript Language (stencil/paint)
imaging model	pixel based	high-level graphics
network protocol	fixed	application tailored
extension mechanism	static	dynamic
hardware access	hardware tailoring	none/device-independent model

Figure 5-7. Comparing X11 and NeWS at a Glance

5.4.2.1 NeWS (Network extensible Window System)

NeWS, a powerful distributed window system based on a unique synthesis of existing window-system technologies, combines the device-independent, powerful imaging model of the PostScript language with window system and operating system concepts such as canvases, event handling, and lightweight processes.

NeWS 1.1 is available on all Sun-3 and Sun-4 systems. Sun also has a technology licensing program for NeWS, in which the reference source for NeWS is available to other vendors for porting and delivery to non-Sun platforms. At the time of this publication, over 60 NeWS licensees existed, and ports are in progress on such varied platforms as OS/2, MacOS, A/UX, and VMS. Sun's reference source product contains ports for 4.2BSD UNIX, UNIX System V.3, ULTRIX, and the 386 architecture.

NeWS hides hardware specific information from the application. Details such as number of color planes, display resolution, and special-purpose rendering hardware are not visible to the application. This approach makes significant hardware acceleration transparent to the application. However, NeWS does not handle some performance techniques, such as color table animation and raster operations.

The extensibility of NeWS is through its use of the PostScript language as an interface, with enhancements appropriate for an interactive window environment. This enhanced language interface allows application programmers or toolkit designers to actually program the window system. If a facility does not exist, a developer can add it. This capability allows developers to actually build portions of their application inside the window server, for better network use, and for speed and flexibility in controlling the user interface of the application by putting more control of the user interface on the server side. Only high-level events are relayed over the network. For example, if pop-up menus are implemented in the server, the client is not informed that the user is looking at various pop-up menus, only that a given pop-up menu has been selected. This limits the use of the network link and improves user interface responsiveness even when network performance is degraded. Another way of reducing network traffic is to load actual rendering code into the server once, rather than having the client do all the rendering.

The lightweight scheduling mechanism within NeWS implements a single-threaded multitasking environment independent of the underlying machine operating system. A multitasking applications environment can, therefore, be provided on operating systems that do not offer true multitasking, such as the Atari OS and the Macintosh OS.

5.4.2.2 X11

The X Window System, like NeWS, is a portable, network-based window system that addresses the need for application portability.

X has a raster-pixel based imaging model. Although it is device independent, window system allows an application to tailor itself to the hardware it is running on. X uses techniques such as color table animation and color raster-ops. The ability to directly access the bitmap is necessary for applications attempting to push the graphics hardware to the limit. Until there are no graphic hardware-bound applications, this need will remain.

X has a static extension protocol, unlike the dynamic extension of NeWS. This encourages the use of a standard protocol for almost all applications. An application that requires custom protocol extensions requires the user to create a new X server with the extensions compiled in — similar to the process of installing a new device driver in the UNIX kernel.

5.5 Product Transition

Sun is providing two key features for a smooth transition between window platforms. The first feature allows SunView window applications to run side-by-side with applications running on X11/NeWS as they are developed. The second ease-of-transition feature is the similarity between the high-level programming interfaces to each window system.

5.5.1 Binary Compatibility

SunView binary compatibility means that SunView / SunWindows applications and applications based on X11/NeWS can run simultaneously. It is the bridge between old and new window system environments. Users can run their own or third-party applications based on SunView side-by-side with applications based on X11/NeWS or View2. All SunView applications run unmodified in the View2/X11/NeWS environment. Binary compatibility is not a mode; a user simply runs the SunView binary. SunView binary compatibility has some limitations, though the inherent functionality of applications remains unchanged:

- SunView does not support remote windows, unlike the View2 toolkit. Applications based on versions of SunView must be running on same machine as the window server.
- A white stripe (16-pixels wide) surrounds each SunView window.
- SunView windows float on top: No SunView window can ever be obscured by any View2, X11, or NeWS window (or menu, notice, or property sheet).
- Primary selection between SunView applications is with Copy and Paste function keys or menu selections to transfer material to and from the clipboard and to and from SunView, View2, and NeWS applications. There is no secondary selection between SunView and View2.

Binary compatibility allows a much smoother and easier transition for both developers and end users.

5.5.2 SunView to View2 Migration

Developers who have worked to the upper levels of the SunView user interface toolkit (those interfaces described in the *SunView Programmer's Guide*) will find they can easily move their applications to the View2 Toolkit. Application structure is maintained, and conversion should take days to weeks, depending on the fidelity of the application to the guidelines in the *SunView Programmer's Guide*. During the conversion, applications developed with the SunView Toolkit can run without change in the View2/X11/NeWS environment via binary compatibility. Development or conversion of an application with the View2 toolkit provides conformance to the OPEN LOOK user interface for that application. (See the *SunView to View2 Conversion Guide.)*

For more details on Sun's product migration strategy, please see the *Sun Windows Transition Story*.