



The Wavelink Architecture

Enabling Enterprise-Class Applications Over Wireless Networks

Wavelink Corporation • www.wavelink.com
11332 NE 122nd Way, Suite 300 • Kirkland, WA 98034 USA
Tel (425) 823-0111 • Fax (425) 823-0143

The Wavelink Architecture

Enabling Enterprise-Class Applications over Wireless Networks

Background

Enterprises are increasingly seeking to “mobilize” business applications to provide mobile employees with anytime, anywhere access to important corporate information and productivity improving applications. In North America alone, enterprises are expected to spend well over \$500 million by 2002 on wireless applications software.¹ By 2005, global spending on mobile Internet software and services is expected to reach \$9 billion.² In order to maximize the benefits from these significant investments, enterprises should adopt a wireless software architecture that is designed for and specifically meets the needs of enterprise-class wireless application development, deployment, and management.

This document provides a summary of enterprise-class wireless application architectural requirements, different wireless application architectures, and explores why the Wavelink thin client/server architecture is best suited for the requirements of enterprise-class wireless applications.

Wireless Applications Overview

Enterprise-class wireless applications

Wireless applications are either informational or transactional in nature. Informational applications deliver informational content to a device. Examples include stock market performance, sports scores, weather, and news. These types of applications are predominantly consumer-oriented and do not typically exhibit any significant information exchange with a corporate database.

Transactional applications involve user input to a database and are especially common in enterprise applications. Enterprise applications range from business process applications to widely used personal information management (PIM) and e-mail applications. Examples of business process applications include exchanging information with a sales force ordering system, making entries into a customer support database, and various supply chain oriented applications such as receiving products at a loading dock, stocking products in a warehouse, and taking physical inventory in a retail store. Other transactional applications that require sensitive data exchange include stock trading, bank account transfers, and mobile e-commerce, or m-commerce.

Unlike informational applications, transactional applications require a tight bond between the wireless device, the enterprise system or application, and the actual information exchange to assure that the necessary database updates takes place without loss of sensitive and often mission-critical data. In contrast, if an informational application fails to deliver data to the wireless device, the browser can be used to simply request a re-send of the data.

Understanding the architectural requirements of “transactional” enterprise-class applications will allow businesses to select the most appropriate solution for supporting their wireless applications needs.

¹ Research Portal

² Goldman Sachs

Enterprise Application Requirements

Fundamentally, enterprise-class transactional application requirements can be summarized in two broad categories:

1. **System-based application control:** In order to maintain a tight bond between the application, the device, and the data collection, the application server must maintain complete control of the application flow to assure data exchange integrity. Enterprise wireless application implementations that do not incorporate this design principle will invariably be exposed to data loss, and ultimately fall-short of their objectives to improve business productivity.
2. **Overall system flexibility:** This requirement encompasses two main areas associated with the adaptability of the wireless solution. Enterprises will be faced with the challenges of supporting potentially multiple wireless device types with wireless applications that will be updated or modified frequently over time. An effective wireless solution should be able to address the following issues:
 - **Heterogeneous device support:** This complexity results from the fact that wireless devices come in varying “shapes and sizes.” Device platforms range from legacy DOS-based terminals to PalmOS and PocketPC to emerging J2ME devices. Furthermore, device input mechanisms include a mix of keyboards, pen tablets, and thumb dials, to name a few. An effective solution should assist the IT professional in managing this complexity and eliminating unnecessary re-work to accommodate varying devices.
 - **New application version deployment methodology:** The new version deployment methodology refers to the distribution method for new versions of the enterprise application software. Many applications change frequently due to new requirements or database modifications. In supporting a population of thousands of wireless devices, the IT professional’s challenge is the need to keep all of the devices updated and in the field where users can remain productive.

The various architectural choices can now be evaluated in light of the enterprise-class applications requirements above.

Architecture Alternatives

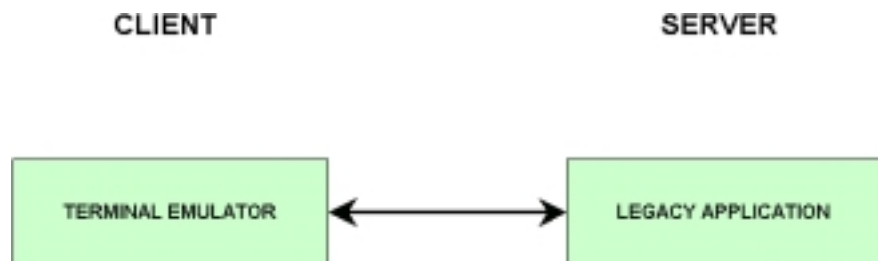
Three basic architectural choices exist for implementing the user interface (UI) component of an enterprise-class transactional application. These alternatives are listed here and will be described in the following section.

1. Terminal emulation
2. Thick client
3. Thin client

Alternative 1: Terminal emulation

A terminal emulator is only possible when a legacy application (3270, 5250, VTXX) is to be presented to the user. This essentially turns the mobile device into a dumb terminal while the legacy application runs on the host. With a terminal emulation program, the client handles the legacy data stream itself, performing all coding and encoding functions according to the rules of the specific protocol.

The following diagram shows the distribution of components in a thin client architecture that uses terminal emulation.



Because the legacy application runs on a host, a terminal emulator is a type of thin client, but one that lacks the common strengths of a three-tiered (controller-based) thin client/server architecture. The *Wavelink Server* section of this document describes these issues in detail.

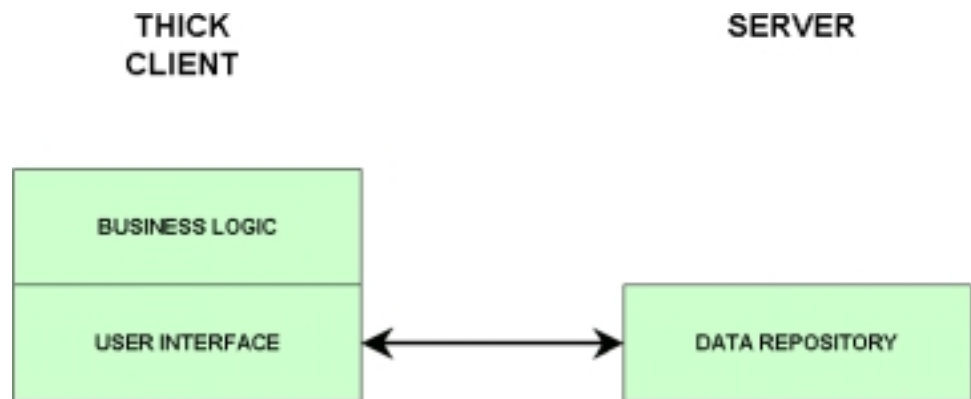
The primary advantage of the terminal emulator is that if the application developer is already conversant with the programming language and idiosyncrasies of the host, sometimes the easiest response to hardware/software upgrades is to modify the host application itself. For example, the developer can change the screen size of the host to match a specific wireless device with which it is communicating.

The primary limitations of this approach include limited flexibility in functionality including support for multiple form factors, built-in support for peripherals, and direct data access from ODBC/JDBC compliant databases.

Alternative 2: Thick client

In a thick client architecture, the application resides on the wireless device and communicates directly with a data repository. Using this definition, the thick client will typically be a custom client that is both device- and application-specific.

The following diagram shows the distribution of components in a thick client architecture.



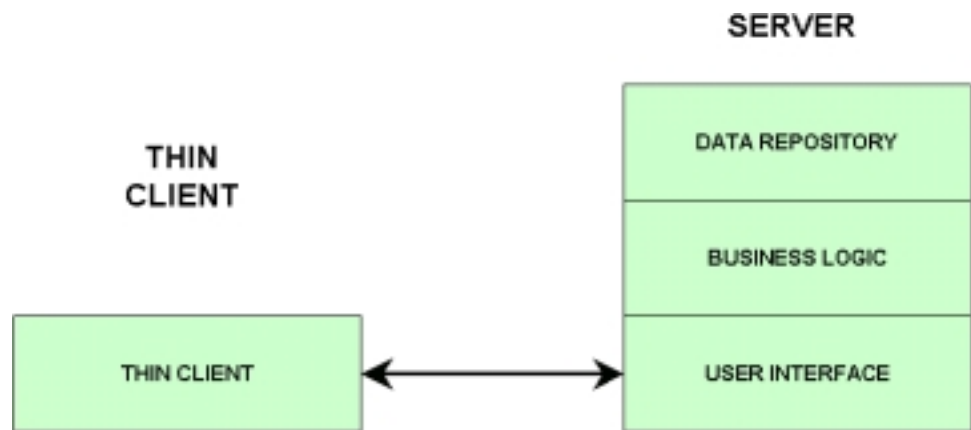
As a result, the thick client lacks flexibility and dramatically increases maintenance costs. All clients in the wireless network must be modified whenever the application changes. This is in contrast to the thin client approach where the business logic resides on the server and is changed in one location only.

Since the thick client is device-specific the client software must be updated or rebuilt whenever any new devices are added to the network, creating additional burdens for the IT professional.

Alternative 3: Thin client

In a thin client architecture, the application's business logic is located on the server. The client resides on the device and performs the functions of graphical display on the user interface and communications with the server-based application. The server software formats the output (screens, tones, etc.) on the individual device type and also interprets input from the device. An application programming interface (API) provides the application interface to the device.

The following diagram shows the distribution of components in a thin client architecture.



Thin clients provide numerous advantages over other architecture types including server-side deployment, redundancy (fail-over support), and other benefits. These advantages will be discussed in several sections of this document.

Thin clients can be browsers (HTML or XML), WAP phones that incorporate a WML-based browser, or the advanced thin client/server architecture that describes Wavelink.

Browsers for wireless devices

This type of thin client can be used on wireless devices that support the following user-interface systems:

- HTML-capable systems
- WML-capable systems (WAP phones)
- XML-capable systems

The strength of the browser lies in displaying colors, graphics, and information. Browsers are specialized for the Web, and as such are the ideal choice for informational, rather than transactional wireless applications.

Transactional applications present special hurdles to the browser because a transactional application is most effective when the application host maintains control of the client-server interaction. A browser is, by definition, a user-directed interface. For example, the user may hit the Back button, type a URL, or close

the application at any point. The browser-based client/server architecture naturally incorporates a design pattern based on its user-directed interface. This design pattern restricts the power of the server. Consequently, the server has no concept of the "state" of an application. If a user exits inappropriately or the browser shuts down, the server simply "hangs," leaving the transaction, database records, and pertinent logs suspended in mid-process.

The generic design of a browser also limits its usefulness in other ways. For example, browsers are not designed to accept input from general external devices. Laser scanners are of particular importance in the wireless world and are usually very difficult to interface to general browser-based clients. In addition, browsers are not geared for specific form factors such as "little screen, big screen, and wide screen." They are also not able to manipulate graphical user interface (GUI) elements such as buttons and pull-downs as they relate to the device native look and feel. Browsers in effect eliminate wireless device capabilities instead of creating an abstraction layer allowing the IT professional to better take advantage of device capabilities. The end result is a very limited user experience when utilizing wireless enterprise applications.

It is possible to work around a few of these inherent limitations through the use of customized browsers and Java server pages. However, this type of customization is typically extensive and complicated. The ease of implementation, long-term usefulness, and basic reliability of a customized browser architecture is severely compromised.

Wavelink's Thin client Architecture

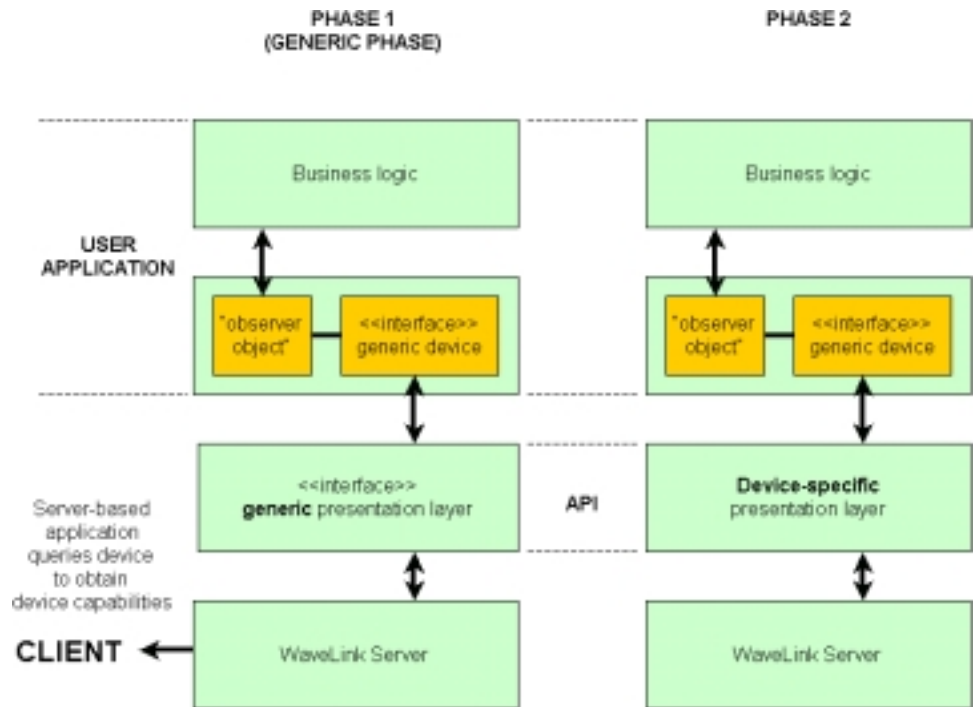
A specialized thin client for wireless devices

Unlike a browser, the Wavelink client is designed to address the requirements of enterprise-class transactional applications. As a result, it conforms to the wireless device native look and feel through enabling the IT professional to utilize all of the device's features as needed. In addition, the Wavelink client functions within a system-directed model where the server maintains complete control over the application flow.

Because the Wavelink thin client is a system-directed interface, the server always knows the application state and how to process user actions; it knows whether a transaction is complete or suspended while the user does something else. This model has two main advantages. First, it protects the system from critical data loss and prevents the server from "hanging" in mid-session. Second, it allows for simpler application design resulting in shorter debug and coding cycles.

The Wavelink thin client architecture further simplifies application design by abstracting the device-specific UI from the business logic. In this design pattern, the server-based application can query the client to see what its capabilities are and then take advantage of its unique features.

As a result, Wavelink's thin client/server architecture enables the creation of "smart" wireless applications – applications that are able to take advantage of device-specific capabilities across a heterogeneous installed base of devices and/or be seamlessly migrated to newer devices and device platforms.



In the Wavelink architecture, the presentation layer is responsible for displaying the state of the application and querying the user for input. The presentation layer shown in the first diagram above is generic (phase 1). The server-based application queries the client to obtain the device capabilities. Once the server obtains the device capabilities, the presentation layer inherits an interface for a specific mobile device (phase 2). The user application remains unchanged and therefore generic. This design pattern yields several significant advantages:

- It allows developers to concentrate on the business logic portion of the application without writing code for a specific wireless device.
- It further reduces the required debug and coding cycles for the application.
- It causes applications to appear just like a native application irrespective of the mobile device with which it is communicating.
- It permits the thin client to use the interface objects of the graphical-based devices (icons, pull-down menus, and other "widgets").
- It enables built-in support for the OS of the connecting device, whether Palm, CE/PocketPC, or DOS.
- It enables built-in support for the input peripherals of the device (scanners, virtual keypads, stylus-based input, etc.).
- It enables built-in support for the form factor of the connecting device (little screen, big screen, wide screen, etc.).
- It enables the use of a common programming language for all devices.

In addition, the Wavelink thin client architecture provides the most effective solution for mixed-device environments which present unique complexity for

application developers. This complexity results from the fact that different models of wireless devices built by a single vendor and models from different vendors interact with and respond differently to the resident client program. These differences can apply to all peripherals including the radio, scanner, display, and input devices. Through the abstraction of the device-specific UI from the business logic, the Wavelink thin client virtually negates the difficulties posed by mixed-device environments.

The Wavelink thin client also incorporates an optimized datastream into its architecture. This benefit, along with the other fundamental benefits of the Wavelink architecture, will be discussed in more detail in the *Wavelink Server* section of this document.

Architecture Comparison Summary

Summary of architecture alternatives and Wavelink's thin client

Specific application and system requirements are summarized for the three client types and the Wavelink thin client in the following matrix:

Architecture Comparisons Across Specific Application & System Requirements

Application Requirements	Terminal Emulation	Thick (Custom) Client	Thin client/ (Browser)	Wavelink Thin client
System-based Application Control	Good	Good	None	Good
Server-based Deployment	None	None	Good	Good
Legacy Application (3270/5250/VTxxx)	Good	N/A	Fair	Good
Direct Data Access (ODBC/JDBC/etc)	None	Good	Good	Good
Multiple Application Support	Good	Good	Good	Good
Built-in Support for Peripherals	Fair	Custom	None	Good
Built-in Support for multiple form factors	None	None	None	Good
DOS Device Support	Good	Custom	Fair	Good
Graphical Device Support (Palm, CE)	Good	Custom	Good	Good
Native Application "Appearance"	None	Custom	None	Good
Reduced Radio Traffic	None	Variable	Poor	Good

*Note: Depends on whether the host-based application is designed for wireless environments and/or multiple form factors

Wavelink Server Overview

Socket-based Connections versus Message-based Systems

As mentioned previously, transactional applications require a tight bond between the mobile device, the enterprise system or application, and the actual data collection. The Wavelink thin client typically uses a TCP/IP socket-based connection, which provides the capability to maintain a continuous connection. (Other protocols supported by the Wavelink thin client are covered later in this section).

The browser client uses HTTP (or WAP, etc) over TCP/IP, which is a message-based connection. Some of the characteristics of each of these types of connections are listed below:

1. Socket-based Connections

- Open, bi-directional “pipe” between client and application
- Messages are fully routable once they arrive at the wired enterprise LAN
- Attempts to connect to same IP Address/Port spawn a new instance and do not intrude on an active application

2. Message-based Connections

- Each message is an independent message that the application sends to a specific IP and Port address.
- Messages must first go to a Web server. The Web server converts them from HTML/HTTP to the application’s messaging protocol
- Load-sharing or fail-over logic must reside in front of the Web servers to provide scalability and redundancy
- Each message from a Client contains identifying information in clear text that can be intercepted and used to hijack the session

In general, message-based connections are adequate for purely informational applications, but are a forced fit for transactional applications. Socket-based connections are natural fits for transactional applications, as well as applications that have both an informational and transactional component.

Wavelink will enhance the socket-based connection approach with functionality that enables wireless applications to maintain operation in areas of poor wireless coverage.

Flexible server capabilities

The server is where the business logic of the application resides and where mapping to the individual wireless clients occurs. System performance and traffic is analyzed and monitored at the server.

Because modern servers support multiple processors, application and device load can be spread among processors within a single server or across multiple servers. This architecture lends itself to enterprise-level scalability.

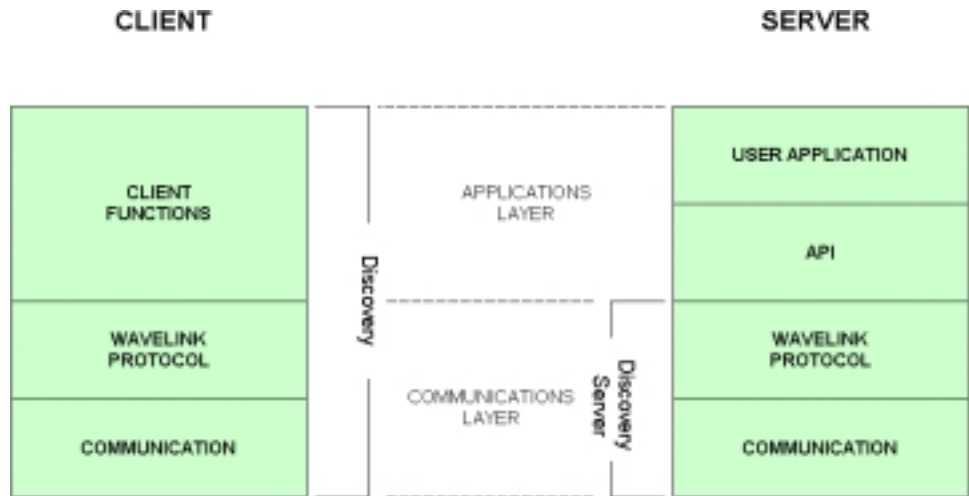
Wavelink supports the following server platforms:

- Windows 9x
- Windows NT/2000
- Unix
- Solaris
- Linux
- Any Java Virtual Machine (JVM) (By Q2 2001)

As noted previously, the availability of a powerful set of libraries (APIs) that normalize the interface to the heterogeneous wireless devices is key to this architecture's adaptability. The Wavelink application interface libraries accommodate more than 30 development environments including the following:

- Java
- Informix
- Sybase
- Oracle
- Visual Basic
- C/C++
- PerlScript
- Visual Fox Pro
- Javascript
- Borland C++
- VBScript
- Powerbuilder
- Python
- Delphi
- Visual J++

Wavelink follows a continuously-connected, server-side model that incorporates an intelligent thin client on the mobile device. (Wavelink is also developing capabilities for maintaining connections only when wireless coverage is unavailable without compromising the application session). The diagram below represents the Wavelink implementation of the standard architecture.



In this model, data travels from the mobile device through the server to the application and vice-versa. Messaging from the application to the server takes place from the user application through the API.

Messaging takes place over a communication protocol that is independent of the transport. The transport is either IP or a network-specific protocol.

Enterprise-class performance features

Wavelink's architecture is designed with enterprise-class performance features:

- **Scalability**

Scalability provides application investment protection, permitting the growth of a wireless network without compromising network efficiency or reliability. Scalability is a built-in feature of the Wavelink architecture.

The Wavelink server is multi-processed, which means that it provides each user with a separate and independent application instance. This design pattern maximizes run-time performance and simplifies coding requirements. Run-time performance is not degraded whether several users or several hundred users are involved.

Because Wavelink is server-based, revisions to the wireless applications are made in one location, eliminating the need to update individual mobile devices.

- **Auto-Discovery**

Through an Auto-Discovery mechanism, the Wavelink Client connects to the server automatically on boot up by specifying:

- A specific network node
- A specific discovery token
- A wildcard discovery token

- **Fail-Over Support**

The mission-critical nature of most wireless application scenarios demands extreme reliability whether the application involves supply-chain automation, database access, financial transactions, or dispatching.

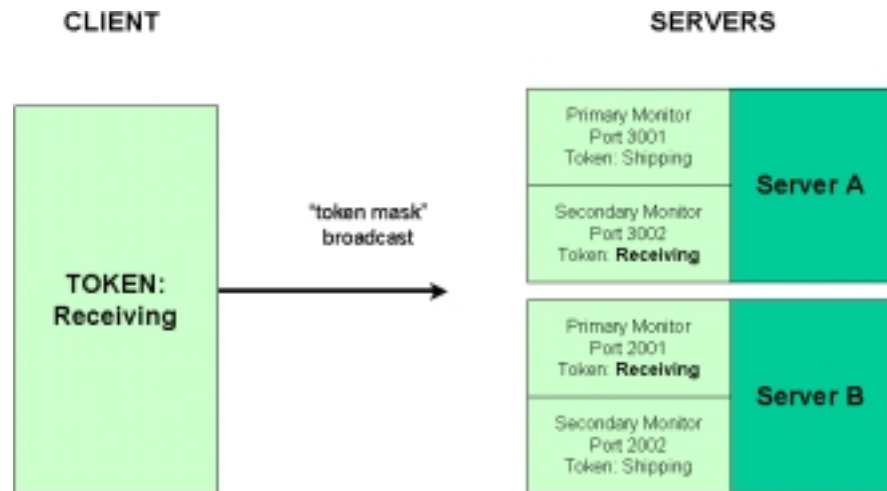
Wavelink provides seamless fail-over support through the optional use of multiple Wavelink servers. Servers can go online or offline without disrupting network communications. Fail-over support is automatic and requires no human intervention.

Fail-over support is accomplished through the Auto-Discovery mechanism. Auto-Discovery makes it possible for a wireless device to connect to one of several possible servers using simple program logic.

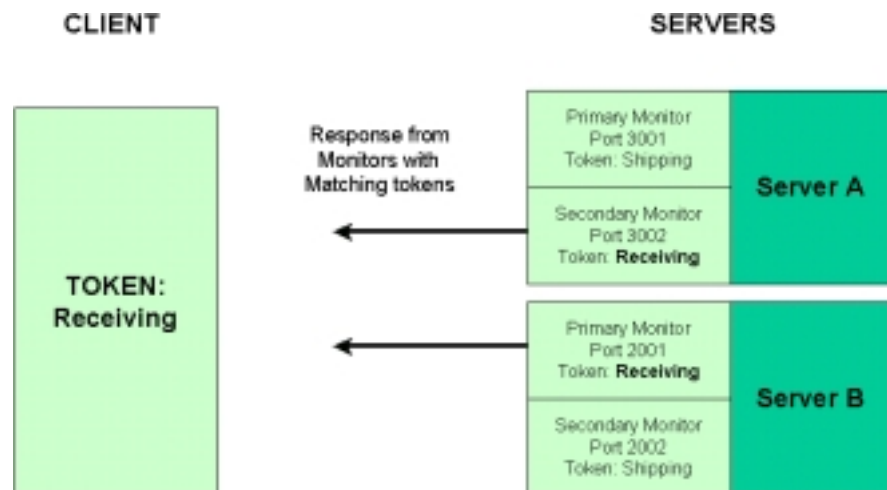
The Auto-Discovery process relies upon the implementation of Discovery tokens. Discovery tokens are assigned to clients and to server sub-components called Wavelink Monitors. The Wavelink Server "listens" for incoming connections on each active monitor that it hosts. In addition to the Discovery token, each monitor also retains an assigned priority (primary or secondary) and a default application.

The following series of diagrams shows the Auto-Discovery process in action for multiple Wavelink Servers on a TCP/IP based network. This process is essentially the same for all network types with the exception of the TCP/IP port assignments.

When the Wavelink Client boots up, it automatically broadcasts its assigned "token mask" to the network (in this case, "Receiving").



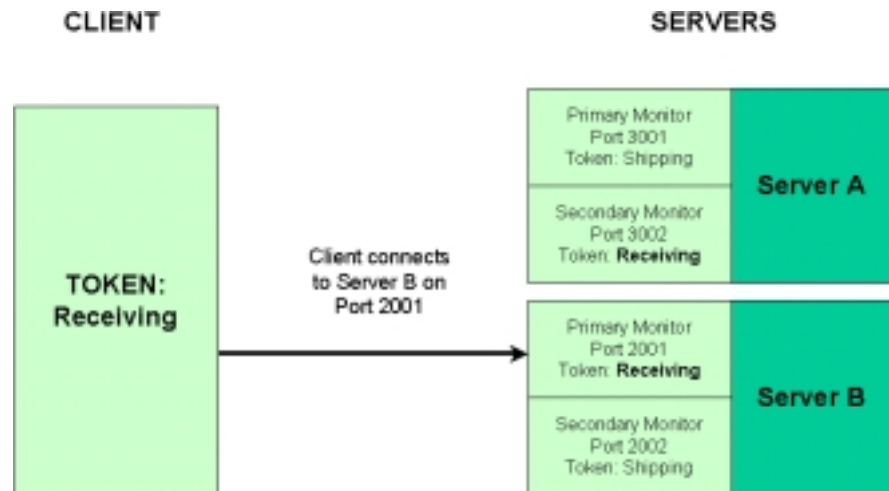
All Wavelink Monitors with matching tokens automatically respond to the broadcast. In this example, two monitors with "Receiving" Discovery tokens respond to the client. The two monitors reside on different servers.



If a "primary" monitor responds within a ten second time out period, the client will connect to it on a "first come, first served" basis. Otherwise, the client will connect to the first secondary monitor that responds before the initial timeout period expires.

Note: This assumes the servers are at equal load status.

The client responds by connecting to Server B on port 2001. The client connects on this port because this Wavelink Monitor is assigned a primary priority.



In this example, if no monitor with a primary priority had responded before the end of the timeout period, the client would have connected to Server A on port 3002.

Note: By default, the Wavelink Client is configured with a wildcard ("*") Discovery token mask, which will return a list of all available Discovery tokens from the network. The user may then select the specific monitor with which to connect.

- **System-Directed Environment Control**

In the typical Wavelink scenario, the duty cycle for the application/server is between 1 and 10%. The application processes user input, responds with a command, and then waits for the user. The application/server processing time is on the order of milliseconds; the user, however, may take several seconds or longer to respond.

Because the Wavelink architecture is of a synchronous nature, no unsolicited data will transmit during the application/server wait time. This provides considerable control over the wireless network, as the mobile device can roam out of range without dropping the connection. Before sending user input, the client buffers the data and verifies its "range" status; if the client is out of range, it displays a message directing the user to move within range. When the user complies, the client automatically sends the data.

- **Security**

The Wavelink Server manages wireless applications and user accounts both on local and remote hosts. The server maintains network security

by enforcing user IDs and passwords. Using a built-in server utility, client start-up menus can be defined and access to applications can be restricted through the use of pre-assigned application groups.

- **Load Balancing**

By Q2 2001, Wavelink will incorporate enhanced performance features such as load balancing to mitigate pressure on each deployed server by distributing traffic during periods of wireless traffic congestion.

The Wavelink thin client/server architecture implements load balancing in combination with the Auto-Discovery mechanism. When the client broadcasts its token mask and Wavelink Monitors respond, the monitor also reports the condition of its host server as one of the three possible states: (1) wide open, (2) close to maximum load, or (3) at maximum load. The programmed logic on the client is simple and efficient; the client will connect, delay, or refuse to connect based on the server state. For example, if the server is close to maximum load, the client will delay connecting to it until the end of the initial timeout period. If no other servers with a "wide open" status respond before the end of the timeout period, the client will then connect to the server that is close to maximum load. (As a side note, the priority assigned to a Wavelink Monitor takes precedence over the server status except when the server reports an "at maximum load" status. In all other cases, the client will connect to a primary monitor before connecting to a secondary monitor).

Summary

Wavelink's architecture – power and flexibility

The specialized Wavelink thin client/server architecture provides the best architecture for transaction-based wireless applications, addressing the needs for enabling enterprise-class applications over wireless networks. This architecture provides *power* in terms of system control for transaction-based applications. It also provides *flexibility* in terms of the variety of mobile devices supported, the range of hosts and databases available to the application developer, and the ease of deployment.

Wavelink Studio's Thin client/Flexible Server Products

Wavelink provides thin clients for Palm OS, Windows CE/Pocket PC, DOS, and Symbol 3000- and 6000-series devices (DR DOS). In addition, Wavelink has successfully completed testing of a J2ME client which will be commercially available by Q2 2001. The Wavelink Studio Application Server is implemented both in C and Java and thus runs on the Microsoft 32-bit Windows products, the popular UNIX systems, Linux, and the Java systems on the IBM Mainframe and AS/400. By Q2 2001, the Wavelink server will run on any Java Virtual Machine (JVM), in addition to the currently supported operating platforms.

Wavelink also provides COM objects to interface to legacy systems and terminal emulators (both client- and server-based) for the legacy systems where the application requirements can be satisfied with a client-resident terminal emulator.

Wavelink, and the Wavelink Studio product family, is uniquely positioned to deliver the best product for implementing transactional wireless applications. Wavelink currently has 5,000 customers and 40,000 site installations in 50 countries