Mobile Data Collection and TCP vs UDP

While most development is typlcally written with TCP, it can present all sorts of problems when working in wireless environments. UDP stands for User Datagram Protocol, while TCP is Transmission Control Protocol. Both are two types of internet protocols. UDP is a simpler than TCP. TCP guarantees delivery of data and does so most of the time. UDP is more of a send and pray protocol. Because of this, implementations utilizing UDP need to handle packet assembly, sequencing issues and dropped packets carefully. UDP presents itself as a good protocol for wireless environments where roaming and power management create issues which bring out the worst in TCP . Moving in and out of range also presents problems when using TCP. The connection will be dropped and it will take some time for a reconnection to happen creating lags in interactive use.

Device Control From None to Extremely High

None and Stateless

Web Based Servers such as Microsoft's Internet Server and Apache's Web Server provide no control over the remote client hardware they run on. They also require a device to be connected 100% of the time. Web based applications typically are stateless as well as forcing work-a-rounds such as cookies and other methods to keep track of state. While control and state may be issues lacking in Web based applications, Server Technologies do provide many powerful pre-canned methods of functionality which can not be ignored.

High Degree of Control - Stateful

Applications developed with Microsoft Embedded Visual C++ Compiler provide direct control over the hardware. Peripherals such as cameras, bar code scanners, fingerprint and RF ID readers are accessed via software written in the C++ Language. Other Language tools such as C#, Java or Visual Basic still force the developer to write software for direct hardware access in C++ and package them in libraries for use. Binary applications provide more control and are generally easier to maintain state in.

Networks and Device Programming Models

The book Building Solutions with the .NET Compact Framework describes how the following development models apply to network connectivity. Loosely paraphrasing from the book these modes are:

Disconnected

A standalone application which may have a local database on the device. This application never connects to a network.

Always Connected

The application is connected to a network and requires the network in order to function. An example of this is a device using an 802.11 or GPRS card to connect to a Web Server. In this case the application resides on the web server. The devices program is Internet Explorer.

Occasionally Connected

The majority of applications fall into this category. In this case the device is connected to a network but must work whether connected or not! This model is a stand alone binary. A local database may be utilized. Synchronization with a remote web server or other server based program will occur when network connectivity is possible. Buffering must be utilized to retain state and data. Δ

IDEs
Visual C++, Embedded Visual C++ 3.0/4.0
VS C#.NET 2003
Java Forte-Netbeans

Sample Solutions

.NET Compact Framework Client

Data can be captured and validated matching an XML schema speciifc to the application requirements. Data is then uploaded as a "transaction" to the target host. Model supports buffering of data during network outages.

SOAP Server - HTTP Client

Via ASP scripting a simple but powerful SOAP server can be customized to provide 7x24 access via IIS. Any client XML enabled browser or application to can perform transactions via HTTP.

Java - RMI-Remote Method Invocation

This is a basic RMI client/server (stub/skeleton) which can be readily deployed from Personal Java to full blown J2EE environments. Used for applications where distributed programming is required. Buffering of data during network outages is supported via serialization. Both Remote calls and callbacks are supported. Environments for mobile clients include: Jeode's Personal Java JeodeRuntime.

UDP Client/Server

Designed for the wireless environment. This model supports data transfer in a peer to peer manner, where nodes can be client, server or both. Model supports buffering of data during network outages.
Logging of data transfer and errors are provided.

mobile 404 307-6731
paulzazzarino@3zwireless.com