Technical Overview of NetMotion Mobility XE

WHITE PAPER

Technical Overview of NetMotion Mobility XE

This document discusses some of the operational and technical details of NetMotion Mobility XE. It is particularly useful to network administrators who require a deeper understanding of how Mobility XE functions before deploying it in their environment. NetMotion Mobility XE and its technology are protected by copyrights and patents both issued and pending, in the U.S. and in other countries.

An Overview of the Mobility XE Architecture

Mobility XE is a highly scaleable, software-based, mobile VPN. It supports both active/active and active/passive high availability, and works with standard network infrastructure including routers, switches, and firewalls. There are two main components of the Mobility XE system: the Mobility server and the Mobility client. They communicate using remote procedure calls (RPC) and the Internet Mobility Protocol (IMP), which run on top of the User Datagram Protocol (UDP).

Mobility Server

The Mobility server manages wireless network connections for mobile devices. It resides on the local area network, on a machine running Windows Server 2000 or Windows Server 2003.

The Mobility server acts as a transport-level proxy for each mobile device running the Mobility client. The server maintains the state of each client and handles the complex session management required to maintain continuous connections to systems that host network applications. When a mobile device becomes unreachable, suspends operation, or moves to a different network, the Mobility server maintains the connection to the client's peer applications by acknowledging receipt of data and queuing requests.

The Mobility server also manages network addresses for the mobile devices. Each Mobility client receives a virtual IP address on the wired network, typically obtained from DHCP (Dynamic Host Configuration Protocol), or assigned from a range of addresses reserved for this purpose on the Mobility server. Mobility also supports the ability to statically assign a virtual IP address to an individual device or user.

Multiple Mobility servers can operate as a server pool with failover and load balancing capabilities. The discussions in this document assume an installation with a single Mobility server. The Mobility server provides tools and metrics that a system administrator can use to configure and manage remote connections and troubleshoot remote connections. A web-based interface allows system administrators to configure Mobility XE settings and manage the server from another PC on the network.
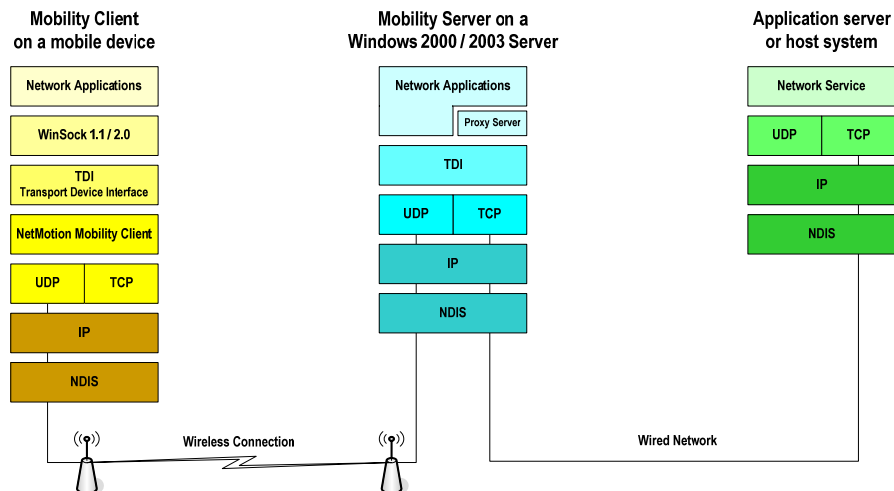
Mobility Client

The Mobility client software resides at the transport driver interface layer (TDI) on supported Microsoft platforms and performs indirection and redirection of application network calls. When an application wishes to use the network, the TDI calls are intercepted, the parameters are marshaled, and the call is forwarded for execution on the Mobility server. It works transparently with operating system features to allow client-side application sessions to remain active when the device loses contact with the network.

A Mobility client device is a standard mobile device or off-the-shelf computer running Windows 2000, Windows XP, Windows Vista or Windows Mobile.

Remote Procedure Call and Internet Mobility Protocols

Mobility XE's Remote Procedure Call (RPC) protocol and Internet Mobility Protocol (IMP) form the technological backbone that connects the Mobility server to each mobile device.

A remote procedure call is a way of allowing a process on a local system to invoke a procedure on a remote system. With Mobility XE, the client's network calls are sent to the server for remote execution. If Mobility operated at the Winsock layer these would be calls such as open socket, bind, connect, send and receive. Because Mobility operates at the TDI layer, the TDI equivalent of these calls is what is sent to the server for remote execution.

The application on the local system is not aware that the procedure call is executed on a remote system. The strength of Mobility XE's RPC-style approach is that it allows the mobile device to go out of range or suspend operation without losing active network sessions. Because session maintenance does not depend on customizing or rewriting applications, off-the-shelf applications will run without modification in the wireless environment.

The RPC protocol is encapsulated by the Internet Mobility Protocol (IMP), which is encapsulated in UDP. The Internet Mobility Protocol compensates for differences between wireline and less reliable networks by adjusting frame sizes and protocol timing to reduce network traffic. This is important when bandwidth is limited, there is high latency, or when battery life is a concern.

Mobility XE also strengthens data security by encrypting all traffic between the Mobility server and clients, and allowing only authenticated devices to connect to the Mobility server. See *Security for Wireless Networks* for more information about Mobility XE security.

Device Registration

When a Mobility client connects to a Mobility server for the first time, the server registers the mobile device's permanent identification (PID) number, a unique number that the client will use for all subsequent connections. This registration occurs only on the first connection and does not require any action by the user or administrator. The identification number is stored in the client system registry and in the Mobility warehouse (LDAP).

The Mobility server stores the PID based on the computer name. As long as the client computer name does not change, the server can restore the PID to the client device even if the client registry is lost. This may happen, for example, if the client device's hard-drive is re-formatted in order to re-install the operating system. When the Mobility client is re-installed and connects, the server searches for a match on the device name. If it finds a match, it will restore the same PID to the device.

Device Connection

When the Mobility client establishes a connection to the Mobility server, it requires the user to authenticate. If authentication is successful, the server and client create the secure VPN tunnel that will be used for the duration of the session.

The mobile worker can use his standard Windows logon credentials to authenticate to the network. The Mobility server authenticates that user against the enterprise's domain using NTLMv2 for native Microsoft authentication, RADIUS authentication, or RSA SecurID authentication. For a Microsoft deployment, a three-way handshake occurs between the Mobility client and server:

- The client sends a list of the supported authentication types. This packet includes the NTLMv2BLOB.

- The server responds with an NTLMv2challenge.

- The client completes the authentication with the response to the challenge.

After authentication, the server and client exchange signed ECC (elliptic-curve cryptography) public keys and related cryptographic materials to perform the Diffie-Hellman key exchange. Symmetric keys are derived from the public keys; these are not transmitted. This applies to all supported authentication methods.

Virtual IP Addresses

Each Mobility client has a virtual IP address on the wired network, obtained from DHCP or assigned from a range of addresses reserved for this purpose on the Mobility server. In addition, static virtual IP addresses can be assigned to specific devices or users. For each active client, the Mobility server relays data directed to the client's virtual address to its current, actual address (the point of presence—POP—address). While the POP address of a Mobility client may change when the device moves from one coverage area to another, the virtual address remains constant for the duration of the user session.

Session Persistence

Unlike IPsec VPNs or SSL VPNs, the Mobility XE VPN does not require a fixed local address. The tunnel is maintained between the Mobility server, which is at a fixed address, and the Mobility client, which can have an ever-changing POP address. By mutual agreement, the client and server maintain the secure tunnel until one endpoint issues a disconnect: this could happen when the user logs off, the administrator quarantines the device, or due to a configurable link inactivity timeout.

The tunnel remains available and application sessions persist in any number of scenarios:

- Suspending operation on the mobile device and later resuming it

- Moving to a different location on the network

- Connecting a mobile device over slow, bandwidth-challenged, or high-latency networks

- Encountering interference from microwaves, stairwells, elevator shafts—anything that interferes with radio signals

- Changing network interfaces (for example, from a WLAN to a WWAN card)

- Moving across gaps in coverage

The configurable timeout ensures that the resources on the Mobility server consumed by inactive sessions are not consumed indefinitely. But in test scenarios, devices have been suspended in the middle of an application transaction, awakened a week later, and the transaction resumed exactly where it left off.

Application Persistence

Network applications are written to use application-level interfaces, such as Winsock (the Windows sockets API). A single call to the application-level API may generate several outgoing or incoming data packets at the transport (IP) layer. In existing mobile networks, if one of these packets is lost, the state of the entire connection can become ambiguous and the session may be dropped.

The Internet Mobility Protocol (IMP) overcomes these difficult issues. The protocol emulates wireline behavior and handles the reliable delivery of data while dealing with various scenarios, including connections lost in the middle of a transmission and roaming. The logical session is maintained and held open, even if the device becomes unreachable or is suspended.

From the standpoint of the application on the mobile device, the application simply waits until it receives a response unless it maintains its own timeout. Mobility XE accomplishes this whenever the server is unreachable, by setting the operation to a pending state. The Winsock interface then returns the appropriate response to the application, whether it is using blocking calls, asynchronous calls or overlapped I/O. The application encounters the same responses and behavior it would encounter on a wired network.

Managing the Connection State

Mobility does not "test" to see if a device is reachable before sending—it simply sends the required data and looks for a return acknowledgement. If the return acknowledgement isn't received, it will re-send for a number of tries, then back off, conclude the device is unreachable, and re-send at a later time. The Internet Mobility Protocol state machine keeps track of packets sent, those acknowledged, and those that need to be re-sent, preserving the integrity of the entire session.

To determine whether an inactive mobile device is reachable, the Mobility system uses keep-alives: the Mobility client periodically sends frames to the Mobility server. The frequency of these keep-alive frames is user-configurable and may be decreased to reduce traffic on bandwidth-constrained networks. If the Mobility server fails to receive expected keep-alive frames during the configured timeframe, it indicates that the device is unreachable in the server's management console. Keep-alives are sent only in the absence of application or other traffic.

Device Is Unreachable

Mobility's use of RPC preserves the state of interrupted data transfers and holds pending data in queue.
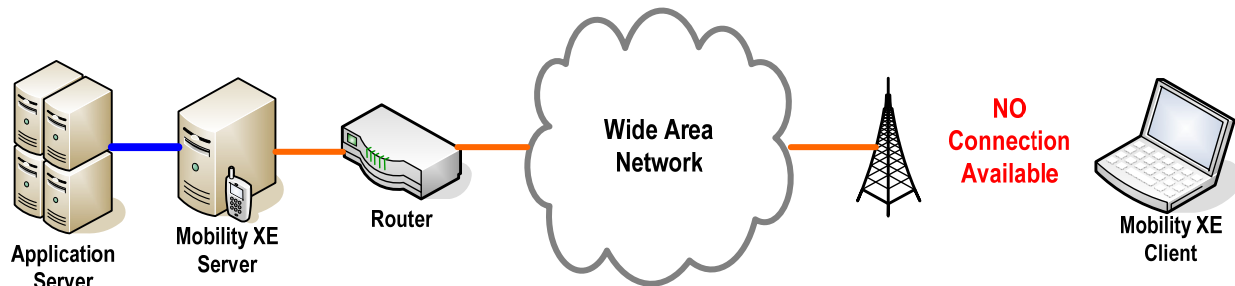
Mobility Preserves the VPN Tunnel, Application & Data even when the client is unreachable

When the application server is transmitting data to the Mobility client and the client becomes unreachable due to a disruption in the network or the suspension of the device, Mobility's flow control algorithms notify the Mobility server's RPC layer to stop accepting data from the application server. The Mobility server's TCP buffers fill, resulting in the TCP window size adjusting to zero (0), which in turn

notifies the application server that data transmission should be paused until the Mobility client is able to receive data. In this state, the Mobility server and application server exchange TCP acknowledgements to keep the connection alive indefinitely or for a length of time configured by an administrator.

When the client becomes reachable again, the RPC layer receives a flow control indication notifying it that the data transfer may continue. The RPC layer then resumes taking data from the TCP stack, causing the TCP connection's receive buffers to empty. When buffer space becomes available, the TCP stack sets the connection's receive window to a non-zero value indicating to the application server's TCP stack that it is again ready to receive data, and the transfer continues.

Device Has Roamed

Mobility uses DHCP to detect and facilitate "roaming"—when mobile devices move across a subnet boundary or to a different network. The user does not need to restart the system or close existing network connections to obtain a new address.

Subnet Roaming

Mobility allows mobile devices to cross subnet boundaries without losing their network connection or application sessions. To do this, the Mobility client must be able to detect that it has moved to a new subnet and must be able to obtain a new point of presence address from a DHCP server.

The Mobility client uses two methods to detect that it has moved to a new subnet. One method is based on information from the network card driver, and is the preferred method. The other method is based on DHCP beaconing, and involves periodic DHCP discover/offer exchanges. Both methods of subnet roaming use DHCP services, at least in part, to perform roaming. The Mobility client uses DHCP services to acquire its IP address and other optional configuration parameters for the new network or subnet.

Roaming Detection in Wireless WANs

On a WWAN, the network infrastructure handles roaming; beaconing is not required. Beaconing is automatically turned off if either of the following is true:

- The Mobility client is connecting over a dial-up (DUN) or PPP (Point-to-Point Protocol) network adapter

- The Mobility client's network adapter acquires a static IP address and does not need to change its IP address in order to roam in the wireless WAN (*e.g.*, roaming is enabled in the WWAN's underlying infrastructure)

Roaming Detection on a Supernetted Network

Mobility provides a setting that modifies beaconing-based roaming detection in a supernetted network. This is a network where two conditions exist:

- A single physical network supports two or more logical networks, and

- DHCP servers assign IP addresses to Mobility clients on more than one logical network

If both of these conditions apply and if the network environment cannot be readily modified to simplify IP address allocation to Mobility clients, enabling supernetted roaming on the Mobility server will improve performance of beaconing-based roaming by preventing frequent attempts to roam when it is not necessary.

Mobility InterNetwork Roaming

A Mobility client can maintain a connection and application sessions when moving between different network media. If network interface cards for different types of network connections have been installed and properly configured on the mobile device, Mobility XE provides application persistence as the client moves between a LAN, a wireless LAN, and a wireless WAN, or any other type of IP-based network. No additional configuration is required on the Mobility client or server.

The operating system and network hardware on the mobile device determine how much user intervention is required for InterNetwork Roaming™. A Mobility client with multiple active network interfaces may be able to switch from one interface to another automatically. For example, if a client device with both WLAN and WWAN cards goes out of range of any wireless LAN access point, communications may automatically transition to the WWAN medium.

On a client device with multiple network interface cards simultaneously active, Mobility XE uses whatever interface the operating system is using. The operating system's interface selection may depend on the order in which the interfaces become active, and may not always result in the use of the "best" interface. The Mobility server modifies the metric for a defined route in the client operating system based on the speed of the network interface, so the mobile device uses the available interface with the greatest bandwidth. If the Roaming—Use Fastest Interface option is disabled, the mobile user may have to manually stop and start interface cards, depending on the operating system.

Sometimes, network connections are available via two interfaces, but the Mobility Server can be reached only via one of these networks. Mobility XE's Client Network Failover allows the client to connect even when the the preferred interface is available, but cannot connect to a Mobility Server. More detailed information on roaming detection and behavior can be found in the Mobility XE System Administrator Guide.

Link Optimizations & User Datagram Protocol (UDP)

There are many behaviors of TCP/IP that make it less than optimal in a wireless environment. To address these, Mobility XE is designed to provide optimum performance over intermittent and bandwidth-challenged network links. Its architecture includes enhancements that allow network traffic over IP to more effectively deal with momentary loss of connectivity from a mobile device, whether due to coverage outages or external factors, such as power management or user intervention. It makes the most efficient use of the given bandwidth using many advanced features that reduce the "chattiness" of transport protocols:

- Selective acknowledgments
- Data and acknowledgment bundling
- Message coalescing
- Reduced and synchronized retransmissions
- Fragmentation optimizations
- Data compression
- Error-reduction algorithms
- Web acceleration

Working in concert with one another, these advanced features provide for the most efficient movement of data. In addition, when "Use Fastest Interface" is enabled (the default), Mobility XE automatically switches to the fastest bandwidth network connection when multiple connections are active.

Below are descriptions of how these features help in a wireless environment, especially a wide-area one.

Selective Acknowledgment Algorithm

When transmitting data over a wide area wireless connection, packet loss can have a severe impact on performance. When either side is transmitting a train of packets, it is quite possible for some of the frames to "evaporate" and never reach their ultimate destination.

Mobility XE takes packet loss into account when sending and receiving data. Using sequence numbers, Mobility can detect whether a frame has been received out of order. If it receives a frame carrying a sequence number greater then the one it is expecting, it alerts the transmitting peer that it did not receive one or more of the frames in the sequence. The transmitting peer then retransmits the missing frame(s).

In contrast, implementations without selective acknowledgment not only retransmit the missing frames, but may also re-send the frames that were correctly received by the peer. These gratuitous retransmissions waste bandwidth and processing resources.

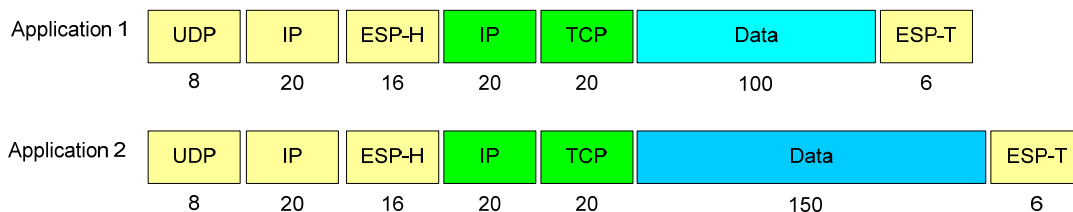## Message Coalescing, Data and Acknowledgment Bundling

Most standard network protocol implementations use acknowledgment frames as feedback to the sender that the peer has successfully received the sent packet(s). The feedback is continuous: every other frame (at most) is acknowledged, generating a stream of small control frames in the reverse direction. These extra frames, though small, can add up to significant overhead.

Mobility XE can reduce the reverse flow of information from every other frame to one in four or greater. It does this by using its selective acknowledgment policy, and adjusting its metrics for determining network latency. This allows more application-level data to be transmitted instead of using the bandwidth for additional control information.

To further reduce bandwidth consumption, Mobility XE uses a message bundling (or multiplexing) technique. With this algorithm, both control and application data from multiple distinct message streams are passed in the same frame. Without this feature, information is encapsulated in its own frame when an application transmits data. As more connections are established, more protocol overhead is required. By bundling these streams together, Mobility allows for more application data to be transferred in the same amount of space, making significantly better use of bandwidth.

Another benefit of this technique is that it allows each frame to carry the maximum amount of payload. In this example, two applications send and receive data in a non-enhanced implementation as follows: Application A sends 100 bytes of data to its peer, and Application B sends 150 bytes to its peer.
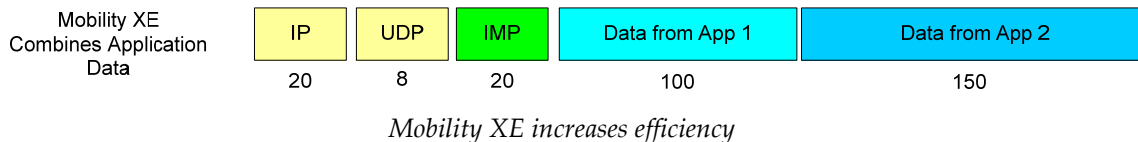
Here is what this looks like on an IPSec VPN (SSL VPNs have similar challenges):

Application 1	UDP	IP	ESP-H	IP	TCP	Data	ESP-T
	8	20	16	20	20	100	6

Application 2	UDP	IP	ESP-H	IP	TCP	Data	ESP-T
	8	20	16	20	20	150	6

Legacy VPNs generate separate frames with protocol overhead for each application

This normally generates two separate frames, one with 100 bytes of data (plus protocol overhead), and one with 150 bytes (plus protocol overhead). Each frame then requires a separate acknowledgment from the peer stating that it was received, so a total of four frames are required to move the 250 bytes of data over the wireless link.

Here's what happens with Mobility XE using the same example: The data from application A (100 bytes) and application B (150 bytes) traverse the link in one 250-byte packet (plus protocol overhead). Only one acknowledgment is generated, so only two frames are required to move the same 250 bytes of data. Mobility XE becomes even more efficient as more applications transmit data.

Mobility XE Combines Application Data	IP	UDP	IMP	Data from App 1	Data from App 2
	20	8	20	100	150

Mobility XE increases efficiency

Reduced and Synchronized Retransmissions

Much of the poor performance over WWAN links can be attributed to retransmission policies that are over-aggressive. Most IP implementations have been tuned to work in high-bandwidth, low-loss environments (like wired infrastructures). But wireless wide area connections experience variable latency, which can easily be misconstrued by normal implementations as packet loss. The transport protocol then incorrectly retransmits a copy of a previous frame.

This retransmission affects performance in two ways:

▪ Duplicate data is forwarded over an already bandwidth-challenged link. This unnecessary transmission consumes the precious resources of a limited pipe and delays transmission of new data and acknowledgments for transmitted data.

▪ When a transport retransmits a frame, it normally assumes that network congestion has caused the loss and employs a back-off algorithm, reducing the total amount of data that can be transmitted at any one time.

By increasing the actual retransmission time when a frame is really lost, the recovery time may be significantly increased, thus reducing overall performance for no good reason. This can result in a spiraling effect, further reducing the transmission of data to an almost a stop-and-wait exchange.

Mobility XE uses many heuristics beyond the standard round-trip calculation to determine network latency. These extra calculations allow Mobility to adjust more rapidly to varying latency and the other conditions that WWANs exhibit. Mobility significantly reduces the amount of duplicate data that is sent, allowing more application data to fit into the communications pipe.

A simple but effective example of these heuristics is Mobility's synchronized retransmission policy. When a frame is transmitted in current implementations, a timeout is set for when an acknowledgment for the frame should be received. If the acknowledgment has not been received, some transports blindly submit another copy of the frame. But when Mobility decides to retransmit a frame, it checks to see if the underlying network layer has processed the previous copy of the frame. If not, it delays submitting another copy. If a mobile device is temporarily out of range, for example, there is no need to send another copy if the previous one has not yet left the local system.

Another example of Mobility's advanced features is its ability to share information about the link characteristics across all application-level connections.

Here's how current transport implementations operate:

- As each application creates a connection to a peer, the transport mechanism determines the characteristics of the link.

- Once this information is gathered, performance increases. But it takes time for enough application data (possibly upwards of 64k) to be passed through the link.

- When the connection is terminated, all the learned information is lost.

In contrast, Mobility XE does the following:

- Mobility retains the learned information; if an application starts another connection, it starts out using the tuned parameter settings.

- When Mobility roams from one network interconnect to another, the new characteristics are applied to all active connections at once. Each connection does not need to go through the exercise of recalculating these values, possibly causing more retransmissions.

Fragmentation Optimizations

The fragmentation of IP packets is regarded by the networking community as a necessary evil that should be avoided. It uses resources in a number of ways, including:

- An intervening system (for example, a router) may have to do further processing on fragmented frames instead of just forwarding them to their ultimate destination.

- It can consume significant resources on the receiving system to reassemble a frame.

- If any part of the fragmented frame is lost, the entire frame must be retransmitted again.

But when roaming from one network interconnect to another (with a possible change in the maximum frame size), fragmentation might be necessary. Here's how Mobility optimizes it:

- There is a maximum message size that can be transferred across a link, and Mobility knows what that size is. If the application submits a request to send data that is too large to fit in a single message, it fragments it before passing it to the underlying network layer. The benefit to this is that the data traverses the network as "normal" (unfragmented) frames and does not cause any extra overhead on intermediary systems.

- The Mobility message fragmentation algorithm has been optimized to ensure that a minimal amount of resources, both computational and memory-related, are consumed to both fragment and reassemble the message. In the event of a retransmission, the fragmentation is reassessed. If the maximum message size actually grew, the frame may be retransmitted in its entirety, again conserving network overhead.

Data Compression

Data compression can improve throughput over "bandwidth-challenged" connections or in congested network environments and save customers a significant amount of time and money (depending on the network billing policies). With Mobility XE, the system administrator can customize compression functionality and determine when it should be turned on, and whether it should be enabled globally (for all users and devices), for a mobile device class, a specific device, or an individual user. Or you can configure Mobility to automatically switch compression on or off based on the current interface speed. Users can roam between high-bandwidth 802.11b LANs and lower-bandwidth GPRS or 1xEV-DO WANs and automatically get the best performance possible.

Mobility XE compresses data that is transmitted between the Mobility server and client. It employs the standard algorithms outlined in RFC 1951 (LZ77 Deflate/Inflate). Only the application payload of each frame is compressed—the transport headers are not modified. This allows Mobility to operate through any policy enforcement equipment, such as firewalls and network address translators (NATs). It also operates over any IP-based network.

Unlike other compression technologies that are associated with specific applications, the Mobility implementation is applied to all application-level data that traverses the Mobility tunnel. No modification or reconfiguration of the application is necessary to take advantage of this functionality. The overhead on the client and server is less than 16K of memory per Mobility connection for dictionary and other housekeeping purposes. It is difficult to predict how much the data will be compressed since it depends on the type of data being sent.

As outlined in RFC 1951: "A simple counting argument shows that no lossless compression algorithm can compress every possible input data set. English text usually compresses by a factor of 2.5 to 3; executable files usually compress somewhat less; graphical data such as raster images may compress much more."

In other words, your mileage may vary. Mobility XE also abides by the "no expansion policy" as defined in RFC 1951: transmitted data will not increase in size.

Since the compression process is computationally intensive, a trade-off must be made to provide the maximum benefit to the user. As a rule of thumb, with end-to-end data rates above one megabit per second, compression may not yield any significant savings. Mobility XE takes this into account and uses other advanced algorithms that detect network latency and compression ratios. Based on the network latency and the percentage of savings gained by transmitting the compressed frame instead of the original, Mobility may elect to transmit a frame in its uncompressed format. This reduces the CPU cycles required to decompress the frame upon reception and provides the maximum benefit to the user. As always, Mobility consumes network and computational resources as efficiently as possible.

Web Acceleration

To speed up web browsing on slow networks, Mobility gives you the option of compressing images. The level of compression is configurable, and you don't have to sacrifice Mobility XE's mobile VPN security (in contrast, the web acceleration solutions provided by most wireless carriers cannot be used in conjunction with a VPN):

- JPEG images are compressed using the JPEG quality metric (the fastest setting results in a file size that is about 28 percent of the original size).

- The number of bits per pixel in GIF images is reduced, which reduces the number of colors. The image is also "flattened": animated GIFs are reduced to one image and textual comments are removed.

Web acceleration is available in two different places:

- Policy Management module: Using policies you can selectively turn web acceleration off and on, change the level of compression, or change the HTTP ports, based on the current network characteristics, on a specific application, or any of the other available conditions.

- Client Settings: Web acceleration is available in the core product without additional licensing. The level of compression can be configured; when on, all HTTP traffic on the designated ports will have the images compressed at the configured level.

Packet Sizes

TCP/IP's packet sizes are not always optimum for wireless transmission. In a wireless environment, error rates rise as transmission power drops, making errors much more likely in marginal coverage areas. Sending large packets in these situations increases the probability that an entire packet has to be thrown away and re-sent. Smaller packets may increase overall efficiency by decreasing the number of re-sends. Many network administrators are unaware that even WLAN's can have substantial numbers of dropped packets and packet errors.

The UDP protocol is much more appropriate for use over wireless networks. It avoids the overhead and inefficiencies of TCP, which was not designed with wireless networks in mind. Instead, Mobility's Internet Mobility Protocol rides on top of UDP, and implements its own methods for dynamically adjusting both packet sizes and timing parameters for the network conditions. IMP, used in conjunction with UDP, handles the far greater variety of transmission speeds and connection conditions encountered in wireless networking. Using this approach also allows IMP to apply its own compression and link optimizations, which can double the throughput over bandwidth-constrained networks.

Reliability

Because the UDP protocol is connectionless, Mobility's Internet Mobility Protocol handles the job of guaranteeing reliable data delivery. It uses its own algorithms for selectively acknowledging packets, handling timeouts, detecting dropped packets and retransmitting them. It is far more sophisticated than TCP/IP—and indeed, it has to be. It not only has to verify delivery amid the uncertainties of a wireless environment, but also has to do so with minimal overhead and with as few retransmitted frames as possible without over-consuming limited bandwidth.

Policy Management

The Mobility server maintains user policies and pushes them out to the Mobility clients in the field where they are enforced. Policy documents are stored in the Mobility warehouse. All Mobility servers in a server pool share a common set of policies.

Mobility policies are pushed to the mobile device when it connects. These policies can be defined to be specific to the device or the user. Policy updates or modifications are applied in real-time after the administrator publishes them for distribution.

For storing, retrieving, and validating rules between client and server, Mobility uses an XML-based rules language. A policy document contains an XML definition of the policy, as an ordered list of rules referenced from the rules documents (a rule or set of rules that control client network behavior). The Mobility server parses the verbose server-side XML into usable code objects, and validates rules against an XSD (XML schema definition) when saving or opening the generated document. When rules are created and saved, it performs an XSL transformation from verbose server-side XML to the client-side format, creating a resource-efficient policy document containing only the information needed by each client.

The policies allow extremely flexible and fine-grained control over user and device access to network resources. While the rules are enforced at the device level, the human-readable rule sets are maintained at the Mobility server.

Policy Enforcement

The basic Policy Management actions that have an impact on connections are: allow, block, disconnect, and passthrough. Mobility enforces these actions at the client. When an application attempts to send over

the network, the Mobility client checks the policy list for the application, port, destination address and other parameters to see if action should be taken. The descriptions, below, assume that actions described are the base action for the policy. Behavior may change if the action is invoked after a session has been activated.

Action	Description
Allow	If the policy action is allow, the Mobility client passes the request to the Mobility server and on to the destination.
Block	If the action is block, the Mobility client sets the I/O status of the connection request at the TDI layer to indicate the connection is pending. The Winsock interface sends a corresponding response to the application that is consistent with the I/O method used. To the application, it appears that the session is still active.
Disconnect	If the action is disconnect, the Mobility client passes an indication to the Winsock interface that the server on the other end requested a disconnect. This makes it appear that the server on the other end terminated the connection, and the session is torn down.
Passthrough	If the action is passthrough, the Mobility client passes the traffic directly on to the TCP/IP transport, so it is passed on as normal TCP/IP traffic, outside the encrypted tunnel between the Mobility client and server.

Policy Management makes granular management of wireless bandwidth, security, and mobile productivity — even over networks the administrator neither owns nor controls — straightforward and achievable.

Quality of Service & DSCP

Mobility XE implements sophisticated Quality of Service (QoS) via Policy Management and integrates with DSCP [Differentiated Services Code Point]. This support can be crucial to maintaining productivity as workers move from high-speed, high-bandwidth networks, to lower capacity, high latency networks. For example, while connected to the LAN via Ethernet, performance may be just fine for the mission-critical enterprise application, running alongside e-mail, web browsing, and other applications. But on a WWAN, administrators want to prioritize use of the narrower bandwidth, and make sure that a web browser and e-mail client do not use capacity needed by the enterprise application. Other VPNs may allow administrators to shut off non-essential applications.
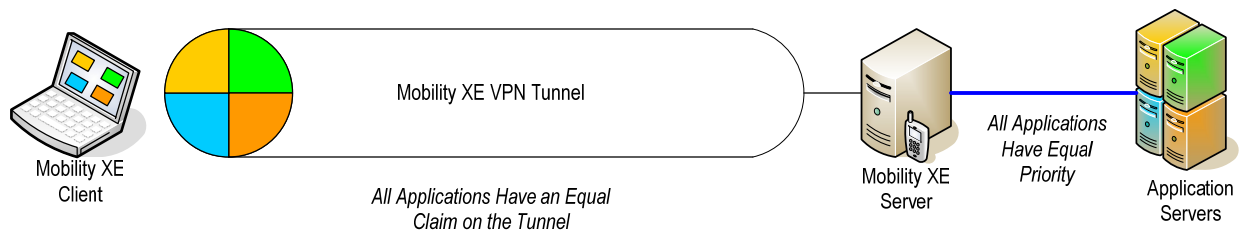
Without QoS: Undifferentiated Access to the Tunnel and Network

Mobility XE allows administrators to specify Quality of Service parameters which are applied between the Mobility client and server, and additionally put DSCP settings which can be used as network traffic

moves beyond the Mobility Server. In particular, Mobility XE allows for different settings based on the network and its characteristics — there can be one set of QoS rules applied to a 1xRTT network, another to EV-DO, etc.
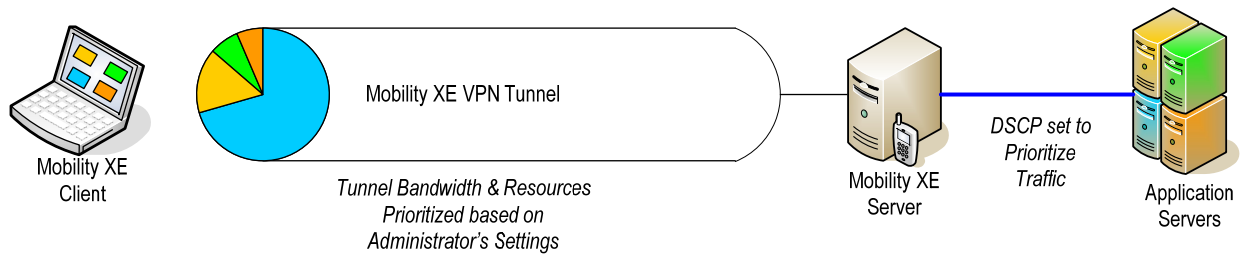
*With QoS: Tunnel and Network priorities set by Administrator*

This allows an enterprise application to always have the appropriate share of network resources, but other, non-critical applications to use whatever bandwidth is available once the enterprise application has used what it needs or is assigned.

Network Access Control (NAC)

The NAC module gathers information on antivirus, antispyware, firewall software, Windows updates and registry, installed files, and processes running on the mobile device. NAC security checks enforced by the Mobility server use this information to assess the health of the Mobility client. For clients that fail a security check, NAC rules provide users with the information they need to bring the client device into compliance.

The NAC module is centrally managed using the Mobility Console. The network administrator creates a 'rule' that can check multiple device attributes. Groups of rules, known as a 'rule set,' are combined to create NAC policies suited to the organization's needs.

Using a 'NAC wizard,' rules or rule sets are established to enforce security policies globally, to workgroups, by class of device, or to individual users and devices. NAC updates are automatically sent to all subscribed users and devices. The NAC rule set is evaluated by the Mobility XE client software at startup, and also re-evaluated every five minutes (this interval is configurable). If a client device fails a NAC policy check, based on severity or corporate policy, the administrator has a number of options for responding and remediating. If the infraction is not serious, they can configure a failure message that explains what the user must do in order to bring their mobile device into compliance. For serious security infractions, the mobile device can be disconnected from the corporate network or quarantined entirely.
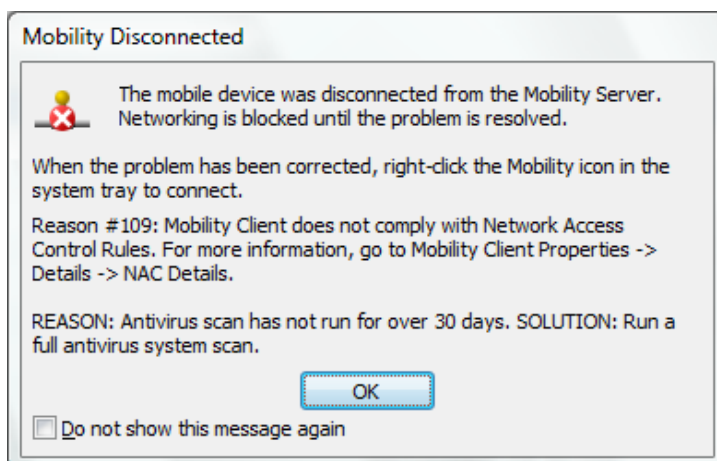
By using the integrated Policy Management module in tandem with the NAC module, an IT administrator can also require that Mobility clients automatically download and install software updates, operating system patches, Mobility XE updates, etc., in order for the device to be considered "healthy" and thus capable of connecting to the corporate network.

NAC Enforcement

Network administrators define rules that are evaluated on each client device with enforcement occurring at the Mobility server. Administrators also determine the severity of enforcement, from warnings, to remediation (using the Policy Management module), to a disconnect or quarantine.

Allow	The Mobility client device complies with NAC policy. Inbound and outbound network traffic allowed via the Mobility VPN through the Mobility server.
Warn	The client does not comply with one or more checks in a rule that causes the Mobility client to display a warning.
Remediate	The client does not comply with one or more checks in a rule that requires remediation. The action required to bring the client into compliance is determined by the system administrator.
Disconnect	The client does not comply with one or more checks in a rule that causes the device to be disconnected.
Quarantine	The client does not comply with one or more checks in a rule that causes the device to be quarantined. The system administrator must clear this state before the device can connect.

Client Activity

The Mobility server collects a number of different details from the active device's session:

- Device Name
- User Name
- Client Status

- Device Description

- Device Class & Device ID

- Server Name
- Virtual Address

- POP Address
- Bytes Sent
- Bytes Received

- Client Registered (Date/Time)

- Connection Established (Date/Time)

- Total Connect Time
- Client Version

- Client OS
- Client (docked/undocked)
- Power source (battery/external)

- Battery (power percent)

- Wireless AP SSID

- Wireless AP BSSID
- Apps using the network (bytes sent/received)

Upon initial connection, the Mobility client sends the session details to the Mobility server, which populates the information on the session details page of the Mobility Console (for each connected device). Once the state of the connection is recorded on the details page, the information is updated when the information changes or on a periodic basis. For example, the POP address details will only change on a roam event. When the mobile device acquires a new POP address, the Mobility client will inform the Mobility server of the new session detail—no other details will be sent if they have not changed. Information that requires more frequent updating, such as the state of battery life on a device, is updated based on a configurable timer.

Because the Mobility server proxies the application traffic on behalf of each Mobility client, the bytes transmitted per application (process) can be derived from the Mobility server without having to burden the client or the network with transmitting this information.

NAT (Network Address Translation)

Because Mobility makes use of UDP it is not susceptible to core problems common to IPSec VPNs and NAT. IPSec VPNs are forced to implement NAT-T (NAT traversal—see RFC 3947), which encapsulates the IPSec ESP packets in UDP in order to traverse firewalls or routers that are not NAT-friendly. Because Mobility is NAT-friendly by design, it does not require the additional layer of encapsulation and overhead required to traverse the intermediary nodes between the Mobility client and server.

Related Information

Additional information can be found on the NetMotion Wireless web site, http://www.netmotionwireless.com and in the System Administrator's Guide.

DuraTech USA Inc.   A Certified 8(a), SDB, DBE, SBE, MBE, WBE firm.
6765 Westminster Ave #314   Westminster, CA 92683
Phone: (714) 898-2171   Fax: 866-704-9132
Email: sales@DuraTechUSA.com   www.DuraTechUSA.com