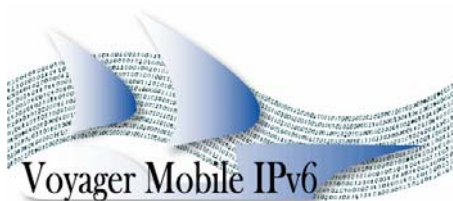


moonv6

Elmic Systems: ***From IPv4 to*** ***Moon V6***

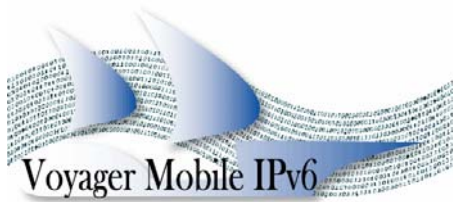


Elmic
systems®
The most fluent way to speak Internet



Agenda

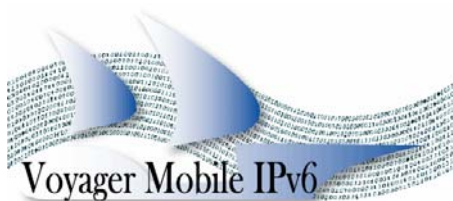
- ◆ Elmic Systems' development of IPv6
 - How Elmic IPv4 became Elmic IPv6 dual stack
 - Lessons learned
- ◆ Elmic Systems and MoonV6
 - Elmic Win32 demo application
 - Elmic interoperability testing
- ◆ Mobile IPv6 scenarios



History of Elmic IPv6

- ◆ September, 1997: development of Turbo Treck TCP/IP starts: a small, high-performance, RFC-compliant, portable “C” source code implementation of **IPv4** built “from the ground up” specifically for embedded systems
- ◆ 2000: ELJP invests in ELUS development of **IPv6**
- ◆ 2001: ELUS starts **IPv6 dual stack**, tightly integrated with **IPsec+IKE**, extending Turbo Treck TCP/IP
- ◆ 9/2002: beta release of Elmic **IPv6 dual stack**
- ◆ 2002: ELUS starts **Mobile IPv6 MN+CN**
- ◆ 2003: ELUS completes **IPv6** development





Size of Elmic IPv6 Effort

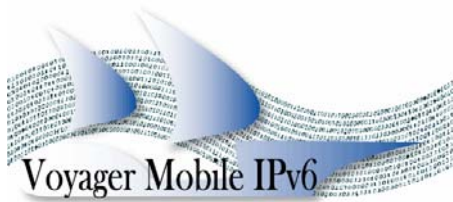
- ◆ Approximately **1000 new functional requirements** to implement for IPv6 dual stack, Mobile IPv6 MN+CN:

[RFC2461].R7.3.3:50

The first time a node sends a packet to a neighbor whose Neighbor Cache entry is STALE, the sender changes the state to DELAY and sets a timer to expire in DELAY_FIRST_PROBE_TIME seconds.

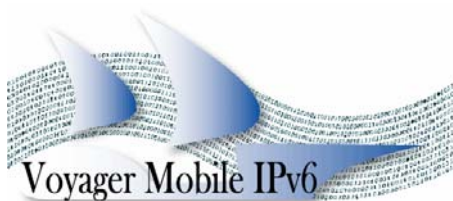
- ◆ Many new RFCs and draft RFCs to analyze and implement:

RFC-1886, RFC-1981, RFC-2373, RFC-2428, RFC-2460, RFC-2461, RFC-2462, RFC-2463, RFC-2464, RFC-2472, RFC-2473, RFC-2507, RFC-2553, RFC-2710, RFC-2711, RFC-2874, RFC-2893, RFC-3056, RFC-3152, *draft-ietf-ipngwg-addr-arch-v3-07, draft-ietf-ipngwg-scoping-arch-04, draft-ietf-ipv6-dns-discovery-06, draft-ietf-mobileip-mipv6-ha-ipsec-06, draft-ietf-ipngwg-icmp-v3-02, draft-ietf-mobileip-ipv6-24, and others...*



Elmic IPv6 Learning Curve

- ◆ Format of IPv6 addresses: 2001:480:261:10:80a1:eaff:ff57:47d3
- ◆ Link-local scope addresses (multi-homing), used for control messaging
- ◆ No IPv6 header checksum, reliance on link-layer CRC and ULP checksum
- ◆ ICMPv6 uses pseudo-header to compute ULP checksum
- ◆ IPv6 extension headers
- ◆ Use `sockaddr_storage` in all dual stack public APIs
- ◆ Prefix discovery uses 64-bit interface ID for address auto-configuration
- ◆ Neighbor unreachability detection
- ◆ IPsec/IKE integration with IP is quite difficult
- ◆ Mobile IPv6 mobile node is very complex



Increase in Product Size

- ◆ Before addition of IPv6 (IPv4-only):

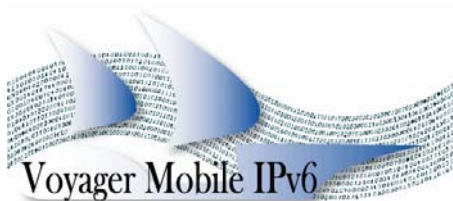
Files:	175
Functions:	1155
Lines:	134544
Lines Blank:	9796
Lines Code:	81357
Lines Comment:	47982

- ◆ After addition of IPv6 (dual stack):

Files:	230	(+ 55, 31%)
Functions:	1481	(+ 326, 28%)
Lines:	214049	(+ 79505, 59%)
Lines Blank:	17275	(+ 7479, 76%)
Lines Code:	128752	(+ 47395, 58%)
Lines Comment:	76359	(+ 28377, 59%)

- ◆ This does not even account for IPsec/IKE!

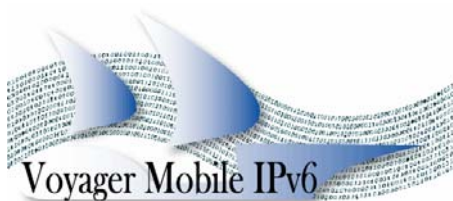




Lessons Learned

- ◆ Development team size: we had exactly the right size (5 design engineers, 5 test engineers).
- ◆ Requirements analysis: a master requirements specification was very helpful for being able to understand the scope of and manage this project. It also helped that the team lead was able to spend significant time up-front doing analysis/scoping.
- ◆ Testing: TAHI made a big difference with regards to our ability to ship a quality IPv6 product on time to our customers. Interoperability testing was also helpful.
- ◆ Development of automated test suite per eXtreme Programming for Elmic dual stack has also worked out well for improving overall product quality.
- ◆ Win32 demo: why develop/test on an embedded target when you can instead run the code on Win32 and use Microsoft Visual C++ with a user-friendly GUI?

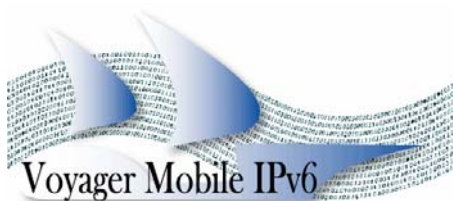




Elmic Win32 Demo Application

- ◆ Free download: <http://www.elmic.com/TreckDemo.zip>
- ◆ Win32 port (preemptive threads) of Elmic IPv6 dual stack that we used at MoonV6 to demonstrate Elmic interoperability with other vendors
- ◆ Has many nice diagnostic capabilities and features, including:
 - User-friendly GUI
 - Auto-generation of tcpdump-format packet capture file
 - netstat view of sockets, routing table, Neighbor Cache, etc.
 - UDP and TCP echo servers, in addition to dual stack server applications: FTP, TFTP, Telnet, Web Server
 - Dual stack client applications: Ping, generic UDP client, generic TCP client, SMTP client
 - DNS stub resolver
 - IPsec/IKE fully configurable from flat-file





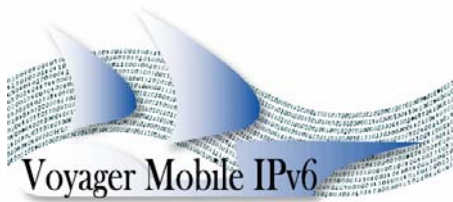
Elmic MoonV6 Testing



moonv6

- ◆ Focus for first couple of days at JITC was on establishing IPv6 routing infrastructure. Since we have a “host” implementation of IPv6, we participated as a “traffic generator”.
- ◆ We assisted in confirming other vendor implementations of IPv6-over-IPv4 configured tunnels.
- ◆ We demonstrated our IPv6 application interoperability (FTP, TFTP, Web Server, Telnet, SMTP) with other vendors.
- ◆ We successfully demonstrated Mobile IPv6 with two other vendors. This testing was done from JITC (MN and HA) to UNH (CN).
- ◆ We assisted in confirming other vendor IPv6 implementations of IPsec/IKE: tunnel mode, transport mode, AH, ESP, all mandatory and many optional algorithms, PFS
- ◆ We used Microsoft Visual C++ to debug IPv6 interoperability issues real-time with our Win32 demo.

Elmic
systems[®]

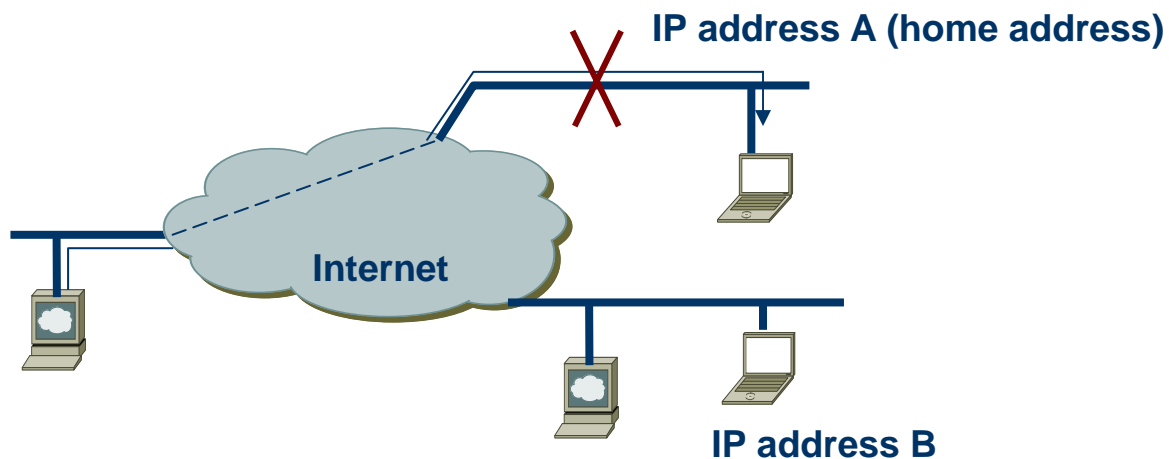


Mobile IPv6 Scenarios

- ◆ Network “push”: accept incoming VoIP packet-switched phone calls on your global scope IPv6 (unchanging) home address, regardless of where you are physically attached to the network
- ◆ Roaming between different L2 technologies: seamlessly switch between different network interfaces (i.e. move from 3G wireless or GPRS to 802.11b) while preserving your application connectivity

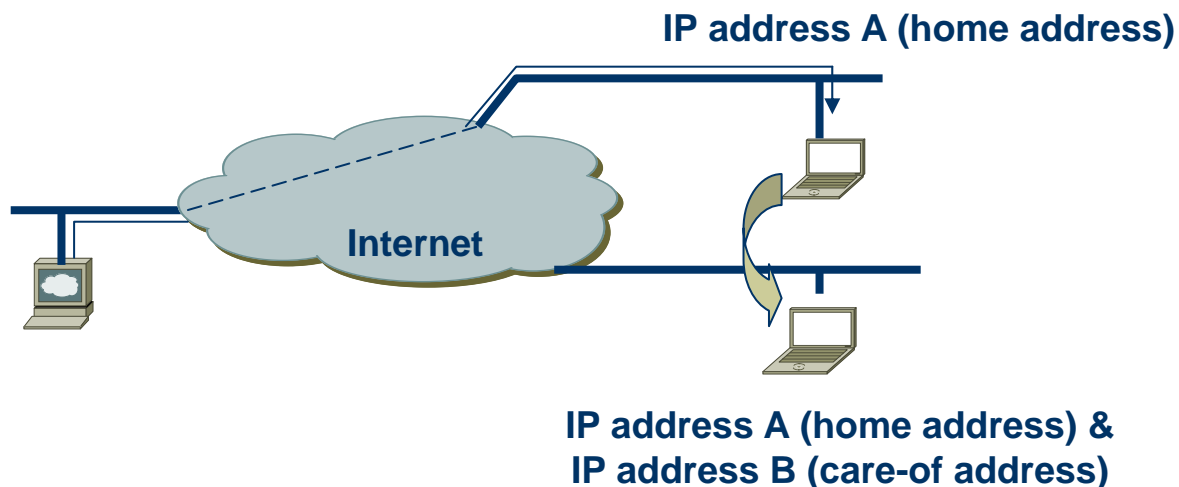
Roaming without Mobile IP

- ◆ Without Mobile IP, devices must tear down and set up connections as they move from location to location

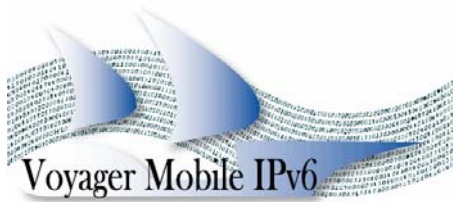


Roaming with Mobile IP

- ◆ Mobile IPv6 allows an IPv6 host to leave its home subnet, while transparently maintaining all its connections and remaining reachable to the rest of the Internet



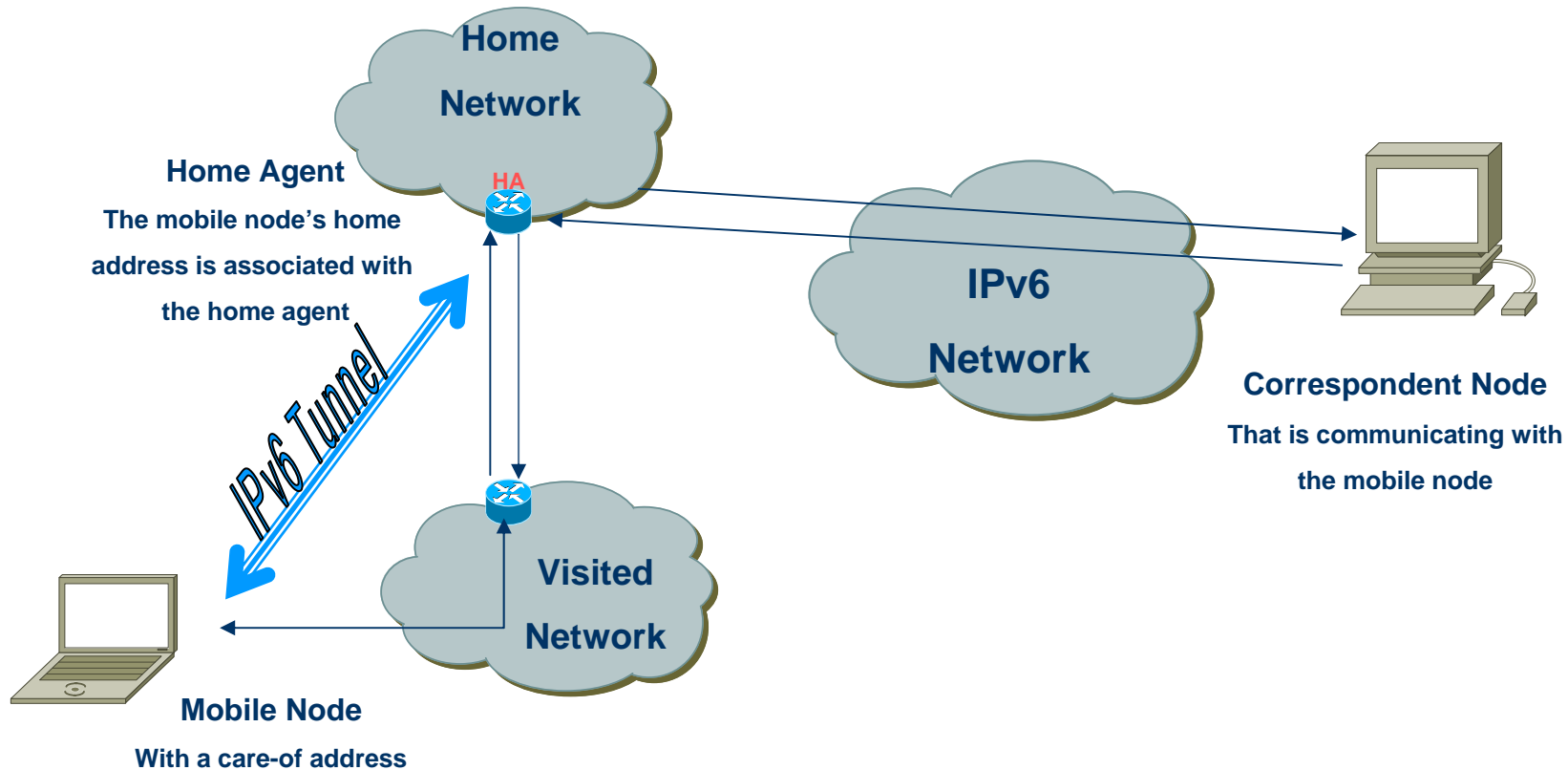
- ◆ Mobile node home address stored in DNS server does not need to change when mobile node moves to a new link



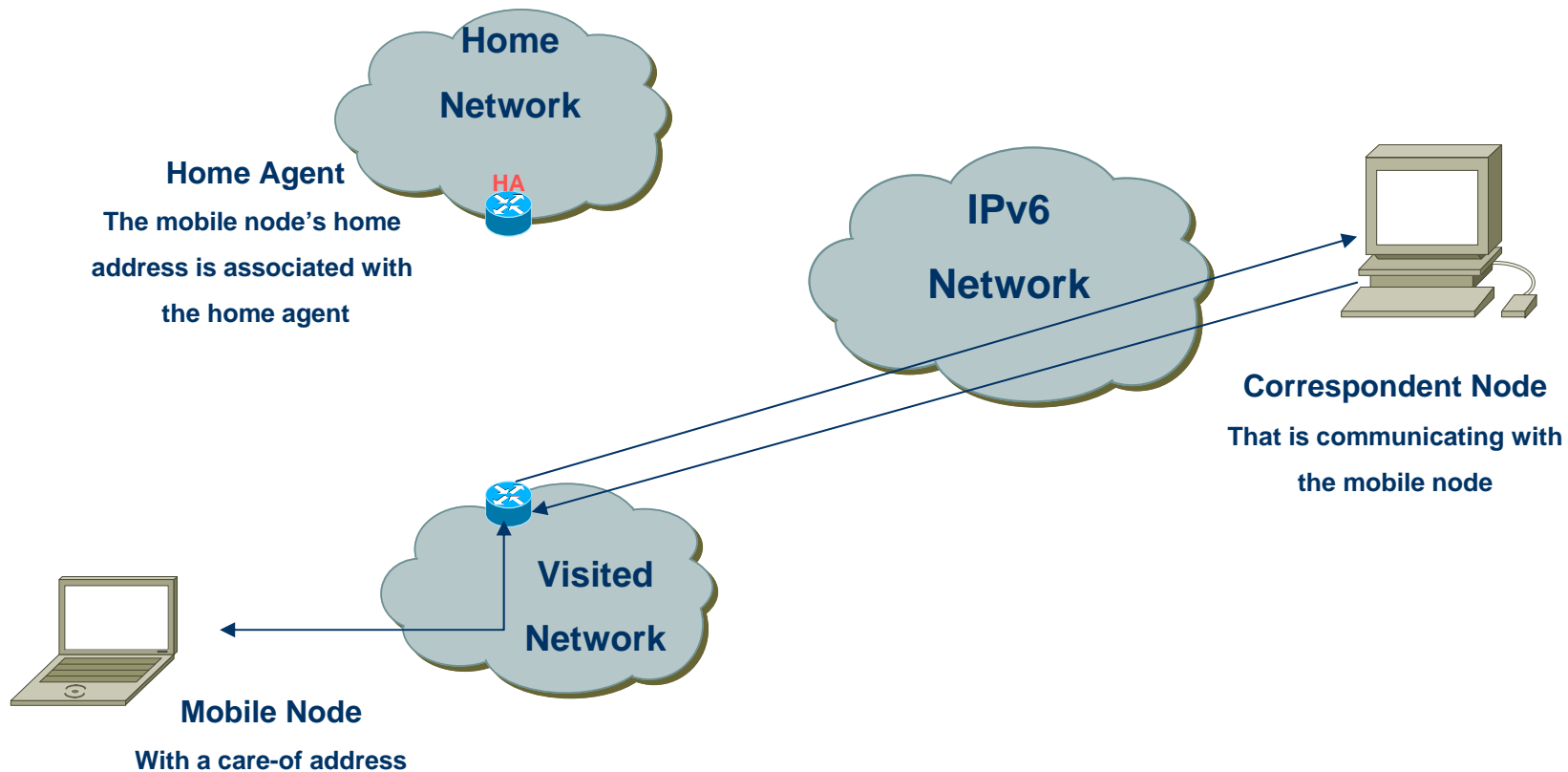
How it works

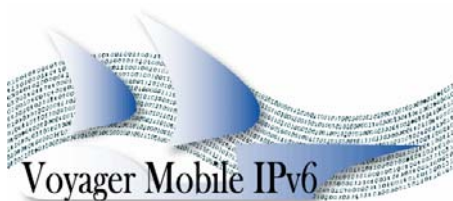
- ◆ Mobile IPv6 identifies each node by its unchanging global home address, regardless of its current point of attachment to the Internet
- ◆ While a mobile node is away from home it sends information about its current location (I.e. primary care-of address) to a home agent on its home link
- ◆ The home agent intercepts packets addressed to the mobile node's home address and tunnels them to the mobile node's current location (I.e. primary care-of address)

Mobile IPv6 Triangular Routing



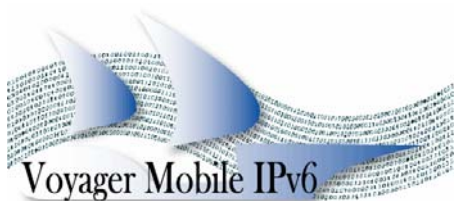
Mobile IPv6 Route Optimization





Elmic Mobile IPv6

- ◆ Mobile Node & Correspondent Node compliant with latest IETF drafts:
 - draft-ietf-mobileip-ipv6-24.txt
 - draft-ietf-mobileip-mipv6-ha-ipsec-06.txt
- ◆ At least an order of magnitude smaller than embedded Linux
- ◆ Route optimization can be excluded at compile-time, further reducing code size; can also disable per application socket
- ◆ MN Binding Update List and CN Binding Cache are fully integrated with existing Voyager TCP/IP performance optimizations
- ◆ Includes Wi-Fi example device driver (Intersil PRISM 2.5)
- ◆ WAP 2.0-compliant Wireless Profiled TCP, plus FACK
- ◆ Supports L2-triggers for MN move detection, as well as Lazy and Eager Cell Switching



Elmic Product Portfolio

- ◆ Elmic System's Turbo-Treck and Voyager Protocol Suites include:
 - ◆ TCP/IP v4 and IPv4/v6 Dual Stack
 - ◆ IPSec/IKE
 - ◆ Mobile IPv4 and Mobile IPv6
 - ◆ Example device drivers, including 802.11b
 - ◆ DHCP/BOOTP Client
 - ◆ FTP, TFTP
 - ◆ Telnet Server
 - ◆ IGMP, NAT, Auto IP
 - ◆ SNMPv1/v2/v3 and SNMP Code Generator



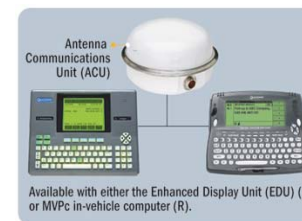
**Wireless Medical
communication Device**



Printers



Set-top-Box

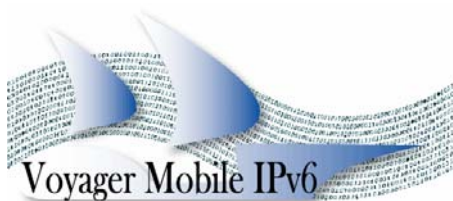


**Transportation
Tracking Device**



Network Storage





Features & Benefits

Features	Benefits
True Zero copy	Provides very high performance numbers and increase in speed
100% RFC Compliance	Provides quality stack
Written Specifically for embedded systems in a scalable and modular design	The code has small footprint and provides plug-and-play functionality
Platform and OS Independent	Stack performs well on any chip or reference design, running any OS
Extensive internal and external testing of the stack	The code is error free
Can run with or without an RTOS	Optimal for small industrial control devices that don't need an OS
Small Critical Sections	Provides a hard real-time, deterministic stack



World Class Support

- ◆ Customer Support
 - Access to network protocol experts
 - 24/7 online support
 - Optional annual support agreement
- ◆ Consulting Services
 - Integration
 - Device driver development
 - Application development

