INSTRUMENTING AN AIRBORNE NETWORK TELEMETRY LINK

Daniel Laird Air Force Flight Test Center, Edwards AFB CA

Kip Temple Air Force Flight Test Center, Edwards AFB CA

ABSTRACT

The Central Test and Evaluation Investment Program (CTEIP) Integrated Network Enhanced Telemetry (iNET) program is currently testing a wireless local area networking (WLAN) in an L-band telemetry (TM) channel to evaluate the feasibility and capabilities of enhancing traditional TM methods in a seamless wide area network (WAN). Several advantages of networking are real-time command and control of instrumentation formats, quick-look acquisition, data retransmission and recovery (gapless TM) and test point real-time verification. These networking functions, and all others, need to be tested and evaluated. The iNET team is developing a WLAN based on 802.x technologies to test the feasibility of the enhanced telemetry implementation for flight testing.

KEY WORDS

iNET, Network, Wide Area Network (WAN) Local Area Network (LAN) Wireless LAN (WLAN), Telemetry Network System (TmNS).

INTRODUCTION

To implement a WAN TM network the iNET team is developing a variant of standard IEEE 802.11b WLAN as an integral component of a WAN. The TmNS is both a program and a networking model that conveniently partitions as a WAN of three basic LANs: the vehicle network (vNET), the ground network (gNET) and the RF network (rfNET) that links them. The rfNET team is currently tasked to test the iNET RF link and the serial TM stream for 'gapless' connection. In this paper we discuss instrumenting rfNET and bridging WLAN and WAN into one seamless network. We begin with a brief description of the TmNS topology and the Open System Interconnection (OSI) protocol model of the International Organization for Standardization (ISO). We then describe the wired and wireless network configurations we used to implement the system wired-wireless link. Finally, we describe the instrumentation required for characterization of

the RF channel and testing of the command and control of instrumentation, primarily at the RF interfaces, i.e., utilities and applications in the Linux gateways.

TmNS TOPOLOGY

The iNET topology is that of a WAN with rfNET connecting the vNET to a gNET interface (cf. Figure1). The network links via RF gateway nodes operating in the independent (or ad-hoc) mode, i.e., all nodes are free to communicate without a central Access Point, often called a mesh topology. The gNET interface is controlled from the test conductor (TC) Windows® XP workstation (TCXP, Figure 1). The vNET test control resides primarily in the FTE Windows® XP system (FTEXP, Figure 1). Each gateway has a WLAN card that controls ad-hoc linking and traffic flow between WLAN nodes of rfNET[1]. The cards are housed in a Linux system that bridges WLAN and LAN via routing tables. Each WLAN configures through Linux boot files, while each LAN configure locally on LAN nodes via the standard Windows® XP network utility. The Airborne Instrumentation Multiplexer (AIM) is a COTS product that serves as an instrumentation IP interface to a solid state recorder (SSR) controlled via RIS Linux OS [5]. One of the nodes in the rfNET is a gateway into vNET and a router steers traffic to and from the FTEXP and other systems. The other node on the rfNET is a gateway into gNET and a router steers traffic to and from TCXP and other systems.

The RF nodes are implemented as Harris SecNet11+ (SN11+) 802.11b WLAN PCMCIA cards. The cards are built on an Intersil™ chip-set that adheres to the open source *wlan-ng* project. This project standardized open source tools which the iNET team exploited to instrument the entire TmNS at the network media layers of the RF nodes. The TmNS topology is that of a wireless mesh i.e., the rfNET nodes are independent centers of wireless communication (Figure 1).

[1] WLAN signals are translated from ISM to L-band via transverters (TVR); see [4].

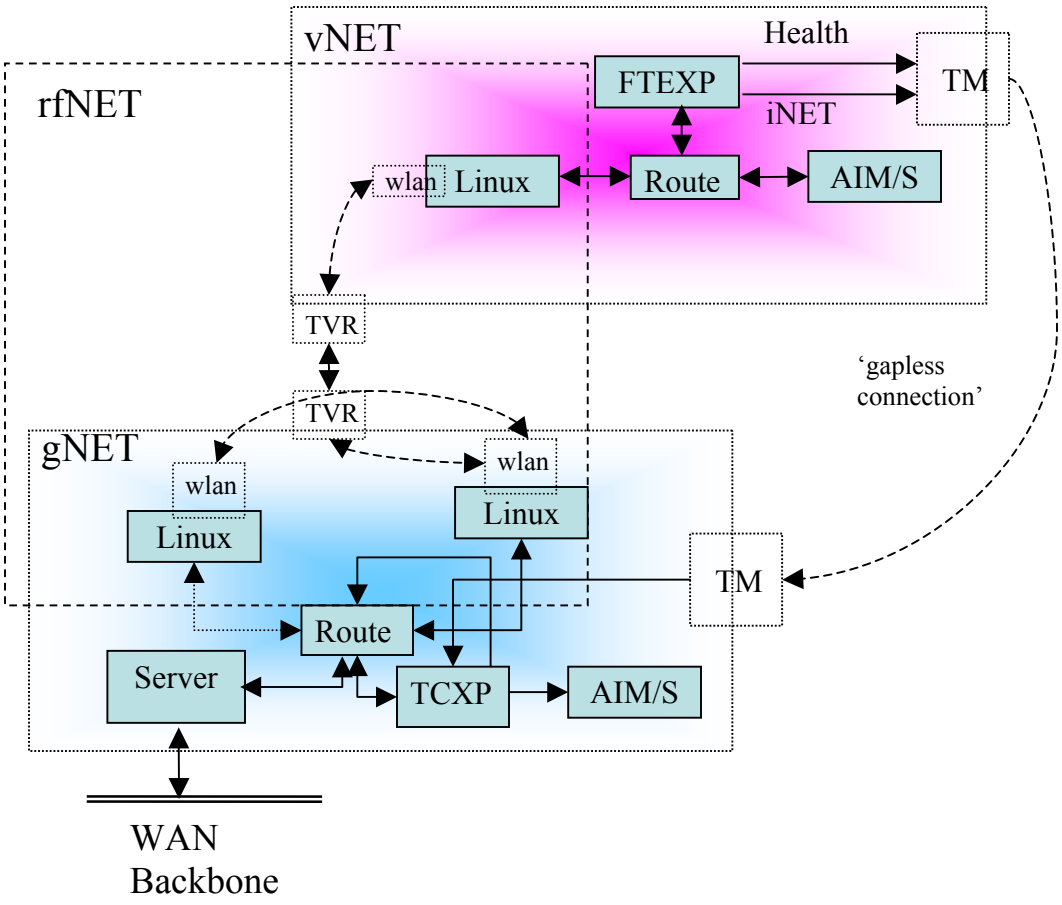


Figure 1 TmNS Block Diagram

OSI Model Overview

Networking is not new, but it is new in the TM community; therefore we begin with a brief description of the OSI seven-layer communication protocol stack [1,2]. One partition categorizes the layers as Host and Media: the host layers process network data segments extracted from the underlying media layers. The extraction is realized as a stack of protocol exchanges that refine physical data into application data (Figure 2, [6]).

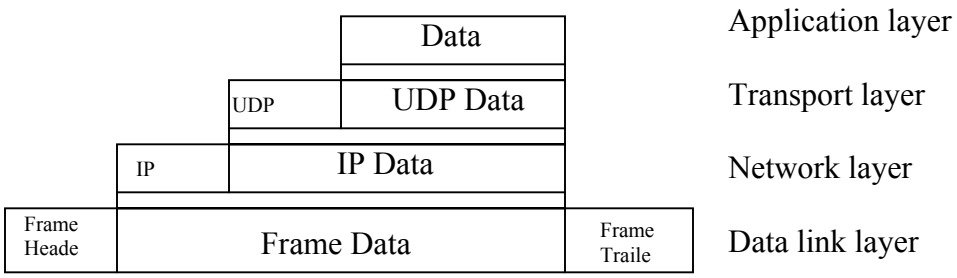
Host Layers

- The **Application Layer** (AL) interfaces directly to application processes and user interface. The common application services provide semantic conversion between associated application processes. Common applications include the familiar Telnet, simple mail transfer protocol (SMTP) and file transfer protocol (FTP).
- The **Presentation Layer** (PL) relieves the Application layer of concern regarding syntactical differences in data representation within the end-user systems. An example of a presentation service would be the conversion of an binary coded file to an ASCII file.

- The **Session Layer** (SL) manages dialogue between applications; it provides for checkpointing, adjournment, termination, and restart procedures. We might view this layer as responsible for configuring and releasing TCP/IP sessions.
- The **Transport Layer** (TL) handles transfer of data between upper layers. The transport layer controls link reliability and tracks packets to retransmit those that fail. This is the TCP or UDP layer.
- The **Network Layer** (NL) provides the means of transferring variable length data sequences from source to destination while maintaining the quality of service (QoS) requests from the Transport layer. The Network layer performs network routing, flow control, segmentation/desegmentation, and error control functions. Routers operate at this level. This is the IP layer. IP is a network layer protocol in the internet protocol suite and is encapsulated in a data link layer protocol (e.g., Ethernet). As a lower layer protocol, IP provides the service of unique global addressing amongst computers.

Media Layers

- The **Data Link Layer** (DLL) provides functional means to transfer data between network entities while detecting and possibly correcting transmission errors from the Physical layer. The addressing scheme is physical, i.e., addresses (MACs) are coded into the network interface cards (NICs) by the manufacturer. The DLL is a composite of the MAC and a logical link control (LLC) layer. The LLC connects or mediates between MAC and NL; controls frame synchronization, flow control and error checking. This is the layer of bridges and switches.
- The **Physical Layer** (PHY) is the electrical/mechanical device layer that carries the data in its raw signal form. The primary services performed by the physical layer are: linkage to communication media, resource sharing among multiple users, e.g., contention resolution and flow control, modulation, signal conversion between equipment comprising the communication channel. RS-485, Ethernet, ATM and 802.11 are common physical layer protocols; hubs and repeaters are examples of physical-layer devices [3]. The PHY is a composite of two sublayers: the physical layer convergence procedure (PLCP) and the physical layer media dependent (PMD) sublayer. PLCP links MAC and PMD. For RF testing we are concerned with the Media layers, for network QoS and throughput we are concerned with the NL and TL, i.e., our concern is with the media layers (1 and 2), NL (layer 3) and possibly NL (layer 4).



Layer	Example Protocols
Application	DNS, TLS/SSL, TFTP, FTP, HTTP, IMAP, IRC, NNTP, POP3, SIP, SMTP, SNMP, SSH, TELNET, BitTorrent, RTP, rlogin, ENRP, …
Transport	TCP, UDP, DCCP, SCTP, IL, RUDP, …
Network	IP (IPv4, IPv6), ICMP, IGMP, ARP, RARP, …
Link	Ethernet, Wi-Fi, Token ring, PPP, SLIP, FDDI, ATM, DTM, Frame Relay, SMDS, …

Figure 2 UDP Stack and Layer Protocols

TmNS NETWORK CONFIGURATION

The TmNS at Edwards is comprised of three networks of four computers, two Cisco routers and two networked GPS time sources using the standard network time protocol (NTP). The three main networks are designated as follows:

- vehicle network, vNET, IP subnet 10,
- ground network, gNET, IP subnet 12,
- RF network, rfNET, IP subnet 11.

There is also a gateway (subnets 192.168.0.0 and 129.198.46.0) to the World Wide Web (WWW). The subnet numbers identify a class C or often called a '/24' network[2]; in our system all subnets are on /24 network 192.168. Table 1 shows ranges and notation for different classes of networks [2, 6].

Address Class	Dotted-Decimal Notation Ranges
A(/8 prefixes)	1.xxx.xxx.xxx through 126.xxx.xxx.xxx
B(/16 prefixes)	128.0.xxx.xxx through 191.255.xxx.xxx
C(/24 prefixes)	192.0.0.xxx through 223.255.255.xxx

Table 1 IP Addressing

Each system has an internet protocol (IP) address, a name and perhaps an alias. A list of all TmNS computers, IP addresses and names is shown in table 2.

IP Address	Name	Alias	Note
192.168.10.50	iNET2	vNETXP	FTE/TM Source
192.168.10.100	iNET3v	vNETLX	v-gateway
192.168.10.200	vGPS	vTIME	NTP source
192.168.10.201	vVLAN	switch	vNET switch
192.168.10.45	DLink	Camera	
192.168.10.170	AIM/SSR		Flight recorder
192.168.11.100	iNET3	rfNETv	v-wireless
192.168.11.150	iNET4	rfNETg	g-wireless
192.168.12.50	iNET1	gNETXP	TCC
192.168.12.150	iNET4g	gNETLX	g-gateway
192.168.12.200	gGPS	gTIME	NTP source
192.168.12.201	gVLAN	switch	gNET switch
192.168.0.1	Server		WLAN gateway
192.168.0.37	Swkstn1		shelter workstation
192.168.0.38	Swkstn2		TM Sink

Table 2 IP Addresses and Names

[2] /24 is an address mask that allows for up to 255 host addresses.

vNET Configuration

Figure 3 is a picture of vNET, the airborne network of the TmNS.

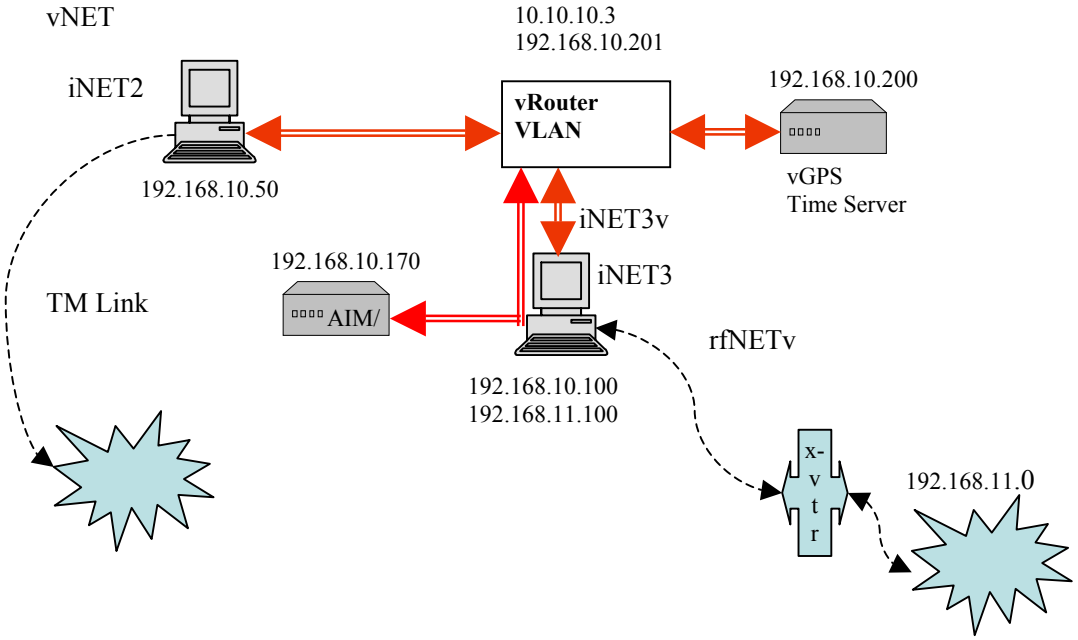


Figure 3 vNET Topology

The FTEXP system is implemented on a Windows® XP/Intel™ computer and named iNET2. The AIM/SSR is a stand-alone unit in vNET and the Symmetricom® Network Time Server (NTS) is also on the RF racks and is named vGPS. This system sources NTP messages to all systems. The Linux gateway is built on Redhat 9.0/Intel™ and is named iNET3, with the wireless side assigned to subnet 11 and the wired Ethernet port assigned to subnet 10, the vehicle subnet. A Cisco 2801 router links all the systems on board the vehicle. This system also contains the TM serial link to carry the source data for gapless telemetry to a complementary system on the ground (Figure 5 below).

gNET Configuration

Although we are only testing an interface to an eventual gNET network, the ground station network topology has similar structure to vNET (Figure 4). The gNET TCXP is also built as a Windows® XP/Intel™ computer system, named iNET1. There is no AIM/SSR system in gNET, as the ground portion of AIM/SSR is a web browser interface that runs on the TCXP system. The gNET also includes a network gateway, named iNET4, also built on Linux Redhat 9.0/Intel™. A 2801 router links all local systems on gNET. The router also includes a gateway to a local server (subnet 0) that connects to the Edwards AFB WAN for access to the WWW. A Symmetricom® NTS sources NTP messages to synchronize all systems on the network.

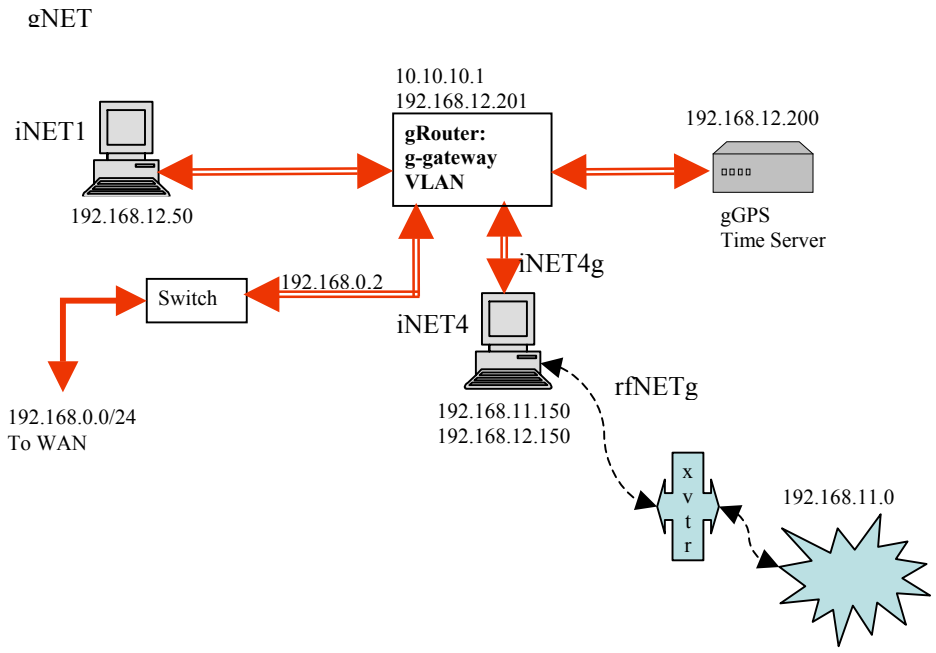


Figure 4 gNET Topology

Gapless Link

The telemetry link (Figure 5) is PCM link that carries the serial stream data from the AIM/SSR for gapless connectivity. The AIM/SSR web page interface resides in Swksn1 (Figure 4.) and all command & control for the gapless data flow is via the Swkstn1 interface.

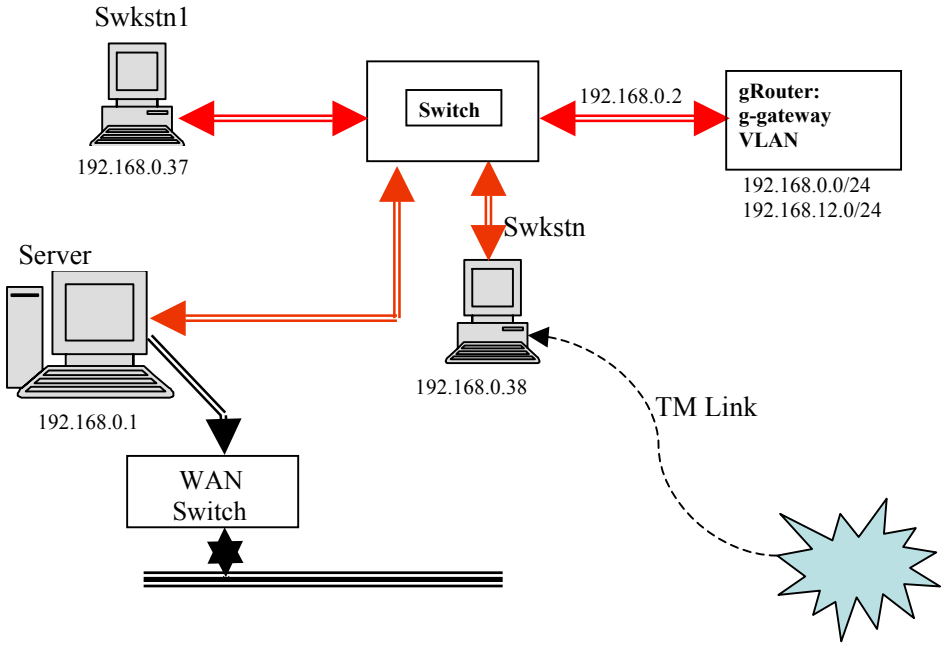


Figure 5 Telemetry Link

rfNET Configuration

Both vNET and gNET have a wired-to-wireless gateway implemented in custom Linux systems built on COTS with Red Hat Linux OS. Each Linux system is configured as the wired-to-wireless gateway into and out of rfNET, i.e., the gateways are 192.168.10.100 – 192.168.11.0/24[3] and 102.168.11.0/24 – 192.168.12.150. The Linux systems also serve as local network routers for all FTEXP, AIM/SSR and NTS. The rfNET distributes over two nodes: two computers built on Red Hat Linux with 2.4.20-8 kernel. These systems are built on rackmount 1U chasses, with Pentium 4 2.8GHz CPUs, 2G RAM and 80G HDD drives, and a DVD/CD reader/writer in each system. The motherboard includes three LAN jacks, one for 10/100 Ethernet and two for Gigabit Ethernet (inactive). The Linux systems serve as wired-to-wireless routers, i.e., as gateways for vNET-to-rfNET and gNET-to-rfNET. The two Linux routers have host names of iNET3 (airborne) and iNET4 (ground). A picture of the entire network connection and IP addresses, with the TM serial link is shown below in Figure 6.

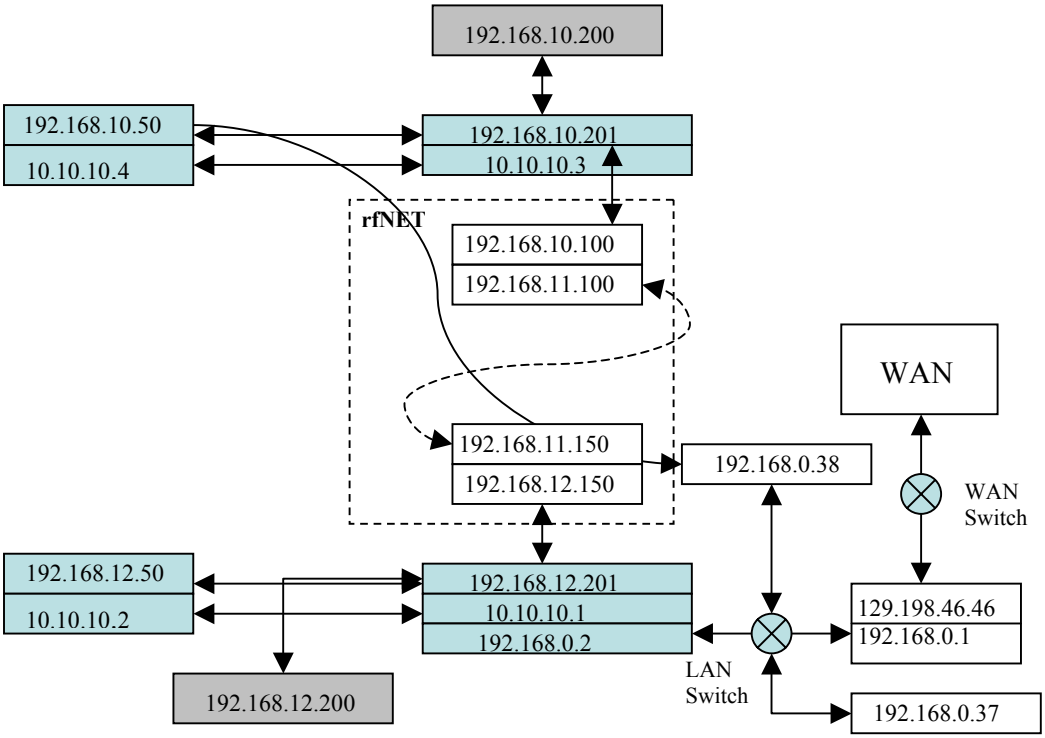


Figure 6 TmNS Linkage and IP Addresses

Network routes were used rather then bridges as the Linux bridging utility could not establish a stable bridge over the PCI-PCMCIA hardware bridge that houses the

[3] The /24 indicates the gateway on subnet 10, address 100 links with the entire wireless '/24' network of up to 255 hosts.

wireless LAN (wlan) card (see below). The fixed routes are assigned automatically on Linux system boot: the system recognizes two networks via the WLAN SN11+ and LAN Ethernet configurations. The routes connect the rfNET (subnet 11) with the gNET (subnet 12) and vNET (subnet10). Additional routes can be added to connect point-to-point hosts if future testing dictates.

The wired side of the Linux routers connects to the vNET and gNET via the local router. The routers are set for simple switching via loadable modules in the router ports. Each router contains two switching modules of 4 ports each. These are extensible and routes can be placed here in future extension of the networks.

The primary hardware of rfNET is the SecNet11 802.11 cards, with linux driver and COTS PCMCIA-PCI hardware bridge. The SN11+ driver is compiled via the source code as a module that also integrates with the PCMCIA-PCI Bridge. The Linux source tree had to be loaded and compiled first. The whole purpose of the Linux system is to link the wireless channel to the wired network and for real-time access to data and monitoring. The 2.4.x kernel was used because newer kernels do not support the SN11+ cardbus socket driver structure.

Router Configuration

Each router has three addressable connections:

1. Router Access (web page)
2. Virtual LAN (switches)
3. Gateway (WAN access)

Once configured, access to the routers is via a web browser connected at FastEthernet connection 0/0. Each router has a virtual LAN (gVLAN, vVLAN) that groups the switch ports into subnets 10 and 12, which can be bridged with a gateway for web access. The gNET gateway links with the WWW server (subnet 0), via local FastEthernet connections (0/1) and the VLANs, to the rest of iNET; this is the WWW access channel for both vNET and gNET. Web access requires persistent routes be established in the routers, iNET3 (subnet 10) and iNET4 (subnet 12) and the (subnet 0) web server.

Each router is configured as a 6-port Ethernet switch with a VLAN IP and a gateway port (0/1) linked to external devices. The g-gateway links with the web server; v-gateway can be addressed to link an ‘external’ network connected to vNET, e.g., another data network. We developed SNMP/MIB (see below) interfaces for the routers, which have many functions available for network real-time monitoring and acquisition ‘snap-shots’.

GPS NTS Configuration

All systems can jam time via the NTS which employs many standard protocols, we use NTP. The Symmetricom® NTS is configured via a serial RS-232 or a network Telnet application interface into the NTS, like all systems on the network the NTS has an IP address and may be queried via SNMP.

INSTRUMENTATION

To instrument the RF portion and the lower levels of the OSI protocol stack we employ open source applications for Linux. The Linux gateway-routers (iNET3 and iNET4) each have several identical utilities, applications and built in scripts to configure the system at boot and during testing such that minimal operator interfacing is required after login. The primary applications for communication link testing are open source products, *iperf* and *ethereal*, custom UDP and TCP source-sink channel set built in LabVIEW™ graphical language and industry standard Manufacturer Information Base (MIB). These source-sink modules establish either a TCP or UDP over IP link over rfNET between iNET1 and iNET2; each is discussed briefly below.

SNMP/MIB and the 802.11 MIB

Industry standards require WLAN cards to expose the MAC via a MIB. A MIB has a hierarchical *tree* structure of *branches* that expose required parameters a *leaves* for specific control and monitoring functions, e.g., network topology (infrastructure or independent), Service Set ID (SSID), channel (10-15), WEP, authentication algorithms, etc. The path to parameters is based on a world wide standard of object identification (OID). The OID is a number delimited by periods that specifies the location in a MIB of measurands, which may be of any numeric or character data type. All MIBs are required to compile in any commercially available MIB browser; with the browsers we gain access to the parameter via the OID call. Browsers usually include display functions for graphical monitoring of dynamical numerical parameters via SNMP and MIB agents (Figure 7).

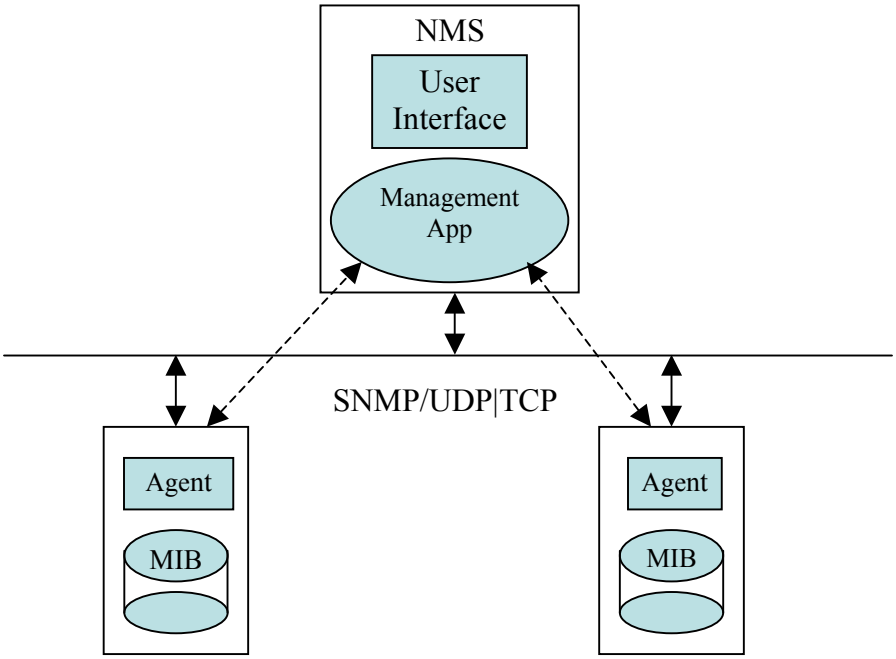


Figure 7 SNMP NMS-Agent block diagram

SNMP is an industry standard protocol for accessing particular MIBs via embedded *agents*, or instrument controllers, which receive commands from NMS, which may reside with the TC and/or the flight test engineer (FTE). Basically SNMP agent service NMS calls to the MIB. The commands are categorized as *get*, *set* and variants *get-next* and *set-next*, which are exchanged over an assigned port; there is also a *trap* assigned its own listening port. The FTE and TC will each have an SNMP application that allows sets, gets and trap monitoring of the AP WLAN. We will exploit the MIB of the router too. We've extended agent capabilities to each WLAN node. This gives the NMS access to the entire network hardware via SNMP applications.

Instrumentation Applications

The link bandwidth testing application we use is *iperf*. This application is best run from the command line, with one node acting as server and the other as client. The command line is included in a supplementary document. This is a user-friendly application used for NL testing; we can monitor the bandwidth and datagram delivery percentages.

Another user-friendly application, with significant power to filter and file data sets is *ethereal* version 0.10.12. This, like *iperf*, is open sourced from the internet. The *ethereal* application is able to capture MAC headers if the SN11+ is configured for RF monitoring using the *libcap* 0.9.3 package to interface with the Intersil® Prism 2.5 chipset on the SN11+ card. The *ethereal* and libcap packages were integrated into Linux as recompiled source code, not as executable binaries. With *ethereal* we can monitor and capture protocol packet payload, with timestamps, time between message receipts, drops and retransmission information. The application allows for packet filter in real-time and during post test processing. This is a great tool for Network Layer and Transport Layer testing.

The LabVIEW™ modules are also simple to use, a client is activated 1st, then a server is activated to generate a message of time samples, the client receives the UDP or TCP packets, strips time and compares the time with local time generating a differential measure that can be saved to file. These modules allow for TL timing between client and server.

Instrumentation Linux Functions

There are several important files in the system that pertain to wlan0 (802.11b wireless card name) and eth0 (wired Ethernet card name) configuration data. Five essential IP configuration files are

1. /etc/wlan/*wlancfg-iNET*
2. /etc/wlan/*wlan.conf*
3. /etc/sysconfig/network-scripts/*ifcfg-wlan0*
4. /etc/sysconfig/network-scripts/*ifcfg-eth0*
5. /etc/sysconfig/network-scripts/*ifcfg-conf*

These are run when the PCMCIA and network services execute. The *wlan.conf* has the parameters to set the network mode to infrastructure or ad-hoc (independent); the file is defaulted to ad-hoc on channel 5 (2.43GHz) of the ISM band. This band is translated to L-band via transverters. To change the card channel you would modify this file. The last two files configure the network interfaces with IP, mask, gateway, network membership, and possibly the MAC address (this is not necessary); this file also contains a command for the OS to configure on boot. The file *ifcfg-conf* handles the route assignments for the Linux gateways.

The Linux routers are set to configure the wireless and wired network routes at boot. The wireless node is a 802.11b 'like' card, which is set for 11Mb/s rate, reduced power and long range linking, with transmit and receive on antenna A (there are two antenna per card: A and B). The function can also be manually run after boot to change the card configuration, the utility is *sn11ctl* and there is a manual (Linux *man*) page that shows the parameters and arguments. We've also written a script that encapsulates sn11ctl and some parameters, leaving the user to enter a command '*sn11conf*' followed by three parameters:

1. *max* or *reduced* (power)
2. *long* or *standard* (range)
3. *t r* (transmit and receive antenna: a, b or diversity).

For example, to establish reduced power at long range to receive and transmit on antenna a, the system executes this command line on boot: *sn11conf reduced long a a*. The SN11+ card configures for reduced power, long range with transmit on antenna '*t=a*' and receive on antenna '*r=a*'. The initialization of the card is via is in another script file, /etc/rc.d/*rc.local*, that includes a similar command line.

There are several scripts and functions built into the Linux OS; for example, to view the 'communication quality' the wireless node we use a function called '*commsquality*' developed from the *wlanctl-ng* open source utility. This function returns the link, signal and noise 'quality values', e.g., *link* 73, *level* 100, *noise* 27. Another important utility to access the SN11+ is *wlancfg*. Like *wlanctl-ng*, this utility has function calls to '*set* and *get'* parameters of the SN11+ card or display the complete configuration and command set. The *wlancfg* has a useful command to get a quick look at the cards MIB and thus one may view the card configuration with the command: *wlancfg show wlan0 all*. There are controllable parameters via the MIB [3], tho all are not implemented by Harris' implementation of the Prism 2.5 chipset.

A useful function we've built as a script is the *wlansniff*, which exploits the *wlanctl-ng lnxreq_wlansniff* utility. This function places the wlan into promiscuous RF monitoring mode and this allows for *ethereal* to capture the MAC header. The command line is *wlansniff* <channel#> <true/false>. The function is in /usr/bin and can be executed from the terminal without specifying the path[4].

Another useful network monitoring tool is an open source utility called *IPTraf*. With this tool we can monitory the IP, TCP, UDP & other protocols throughput and real-time data rates in bits/second and packets/second. The collected information is logged

[4] All system functions are in shell scripts that require no explicit path to execute, just open a terminal and execute the command via the command-line syntax.

for post-test processing. Another useful connectivity testing function is *ping.pl*. This is a *Perl* script that pings remote host, displays connectivity (good/bad) then outputs result to screen and data file. This function will gather data on resynchronization time for dropped rfNET links. A *Perl* script function, *signallog.pl*, reads the output from *wlanctl-ng*, *ifconfig wlan0* and the OS */*proc*/wlan0* file, then writes to both screen and file *signallog*.txt.

To measure message exchange times we use a script that activates a pair of functions *server2way.pl* and *client2way.pl*; this pair of functions bounces a packet between server and client and measures time of travel. We either start server on iNET4 start client on iNET3 or vice-versa. This function will gather data on packet delivery time between vNET and gNET, which have to be synchronized to the same time source for precise time en route returns. We also exploit the embedded Linux SNMP/MIB [3] functions to gather routing information at the wireless gateways.

Media Layer Instrumentation: 802.11 MIB

The iNET team is implementing the rfNET gateway and instrumentation as a real-time Linux network with Harris SN11+ cards, see [4]. The Linux system allows access throughout the entire protocol stack to the node's PHY PMD. Linux configures the WLAN nodes, monitors all traffic (sniffs) and logs relevant captured data via delivered utilities (*sn11ctl*) and open source applications (*ethereal*). To instrument the DLL requires media access and control, therefore we've developed and exploited software (SW) packages to configure the WLAN cards to collect data from the media access control (MAC) sublayer of the DLL and the PHY. The sublayer that connects the DLL and Network layer is the logical link control layer (LLC). This layer is the first hardware independent layer and we may access it via industry standards like *ethereal* and open source utilities like *wlancfg*.

The 802.11 MIB is actually a composite of four basic branches that store most data structures in *tables*, see [3]. The four branches are station management (SMT), the MAC, resources (RES) and PHY. The SMT contains six card management tables, some of these entries we monitor. The RES is the resource branch, which is primarily used for manufacturer specific implementations. We'll not discuss these two branches in any detail here: our concern is with the MAC and PHY. The 802.11 MIB is not accessed via generic SNMP calls, but rather via WLAN programs delivered with the hardware or open source code from the WLAN project, an open source project. The full structure of the 802.11 MIB is beyond the scope of this paper.

In this phase of iNET we are implementing WLAN nodes with Harris SN11+ package. The Harris card, like many others, is built on a Prism II chip-set that controls the MAC and PHY. The MAC exchanges data with the PHY PMD sublayer; the PCLP mediates the transition of MAC packets and the PMD signals. The MAC controls a network allocation vector (NAV) and the interframe time spacing (IFS) which form the timing basis of the collision avoidance algorithms. The MAC-PHY layer exchanges are access via MIB table calls. The SN11+ MAC-PMD controls a direct sequence spread spectrum (DSSS) modulation in carrier sense multiple access-collision avoidance (CSMA-CA) scheme. The CSMA-CA scheme eliminates contention bottle-necks of collision detection schemes such as traditional wired Ethernet. We interface the SN11+ MIB using Harris utilities, standard WLAN and *ethereal*.

Media Access via Linux

The rfNET was implemented on Linux to allow adequate access to the WLAN node MIB. The open source WLAN project was created to develop a complete standards based WLAN system using the GNU/Linux operating system based on the IEEE 802.11 standard. The WLAN contains a device driver and support utility for GNU/Linux supporting the Intersil™ reference design PRISM DSSS WLAN Adapters. This design uses the PCMCIA interface and form factor. The PRISM card is an IEEE 802.11 compliant 2.4 GHz DSSS WLAN network interface card that uses an Intersil PRISM2.5 chipset for MAC and PHY control and function. There are a several tools available in the standard package, the more useful being the *wlanctl-ng* for MIB access and the *wlancfg* for setting and querying the chipset. A resource specific *sn11ctl* program may be used to configure a default setup and the Linux *cardctl* program returns details on the card configuration, status and manufacturer information.

Summary

Our primary interest is instrumenting a seamless wireless-to-wired network to acquire information on a working prototype of the TmNS. We built an rfNET link on Linux and employed open source applications to instrument the traffic routes, primarily on the first four layers of the OSI protocol stack. The vNET and gNET interface prototypes are built on industry standard commercial products, e.g., Windows®, Linux, Intel™, Intersil™, LabVIEW™, Cisco Systems Inc., etc. We exploit the embedded industry standard network instrumentation, primarily open source Linux and Windows® applications, *ethereal*, *iperf* SNMP/MIB and the 802.11 MIB, to gather information on the entire network, primarily at the gateway interfaces.

REFERENCES

1. References: Open Systems Interconnection: Upper Layer Standards and Practices; Baha Hebrawi; McGraw-Hill 1993.

2. TCP/IP: Architecture, Protocols and Implementation; Sidnie Feit; McGraw-Hill 1993.

3. 802.11 Wireless Networks: The Definitive Guide; Matthew S. Gast; Reilly and Associates, Inc. 2002.

4. Airborne Network Telemetry Link; Kip Temple and Daniel Laird; ITC San Diego CA 2006.

5. Vehicle Network Concept Demonstration; Thomas Grace and John Roach; ITC San Diego CA 2006.

6. IP Protocol; Wikipedia: World Wide Web Encyclopedia.