LynxOS Certifiable Stack v2.2 User's Guide

LynxOS-178

DOC-2135-00

Product names mentioned in this document are trademarks of their respective manufacturers and are used here for identification purposes only.

Printed in the United States of America.

Table of Contents

Preface

Typographical Conventions

The typefaces used in this manual, summarized below, emphasize important concepts. All references to filenames and commands are case-sensitive and should be typed accurately.

Kind of Text	Examples
Body text; *italicized* for emphasis, new terms, and book titles	Refer to the *LynxOS Certifiable Stack v2.2 User's Guide*
Environment variables, filenames, functions, methods, options, parameter names, path names, commands, and computer data	`ls -l myprog.c /dev/null`
Commands that need to be highlighted within body text, or commands that must be typed as is by the user are **bolded**.	`login:` **`myname`** `#` **`cd /usr/home`**
Text that represents a variable, such as a filename or a value that must be entered by the user, is *italicized*.	`cat <`*`filename`*`>` `mv <`*`file1`*`> <`*`file2`*`>`

Blocks of text that appear on the display screen after entering instructions or commands	`Loading file /tftpboot/shell.kdi into 0x4000`
	`.....`
	`File loaded. Size is 1314816`
	`© 2017 Lynx Software Technologies, Inc. All rights reserved.`

Keyboard options, button names, and menu sequences	**`Enter`**, **`Ctrl-C`**

Technical Support

Lynx Software Technologies handles support requests from current support subscribers. For questions regarding Lynx Software Technologies products, evaluation CDs, or to become a support subscriber; our knowledgeable sales staff will be pleased to help you. Please visit us at:
http://www.lynx.com/training-support/contact-support/

How to Submit a Support Request

When you are ready to submit a support request, please include *all* of the following information:

- First name, last name, your job title
- Phone number, e-mail address
- Company name, address
- Product version number
- Target platform (for example, PowerPC)
- Board Support Package (BSP), Current Service Pack Revision, Development Host OS version
- Detailed description of the problem that you are experiencing:
- Is there a requirement for a US Citizen or Green Card holder to work on this issue?
- Priority of the problem - Critical, High, Medium, or Low?

Where to Submit a Support Request

Support, Europe	tech_europe@lynx.com +33 1 30 85 93 96
Support, worldwide except Europe	support@lynx.com +1 800-327-5969 or +1 408-979-3940 +81 33 449 3131 [for Japan]
Training and Courses	USA: training-usa@lynx.com Europe: training-europe@lynx.com USA: +1 408-979-4353 Europe: +33 1 30 85 06 00

CHAPTER 1 *Introduction*

Lynx Certifiable Stack Description

Lynx Software Technologies, Inc. has been the premier developer of highly reliable embedded operating systems and software. The Lynx Certifiable Stack (LCS) is a TCP/IP protocol stack that has the features necessary for use in safety critical avionics and other applications.

In the avionics industry, standards such as DO-178B and the European equivalent, ED-12B, establish strict requirements for certification of highly reliable software. LCS is the standalone TCP/IP solution that helps manufacturers to achieve a low- cost, low-risk path to certification and time-to market for applications requiring communications over TCP/IP. LCS is certifiable to DO-178B Levels A and B and is offered as a standalone component for use in a variety of applications.

LCS provides the following benefits:

- The LCS TCP/IP stack is aware of LynxOS-178 partitioning. It is possible for several partitions in the system to use the LCS TCP/IP stack simultaneously without affecting each other.
- The LCS TCP/IP stack uses the POSIX API for communicating via networking and transport layer protocols that makes it possible to use the standard set of networking utilities and APIs for user-level applications.

General

Acronyms and Abbreviations

API	Application Programming Interface
ARP	Address Resolution Protocol
BSP	Board Support Package
CR	Change Request
FTP	File Transfer Protocol
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol
IP	Internet Protocol
LCS	Lynx Certifiable Stack
NIC	Network Interface Controller
POSIX	Portable Operating System Interface for UNIX
SNMP	Simple Network Management Protocol
SNTP	Simple Network Time Protocol
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
UDP	User Datagram Protocol

CHAPTER 2 LCS Software Architecture

Overview

The LCS stack consists of the following components:

- LCS networkingdaemon
- LCS proxy socket driver
- LCS networking controller drivers

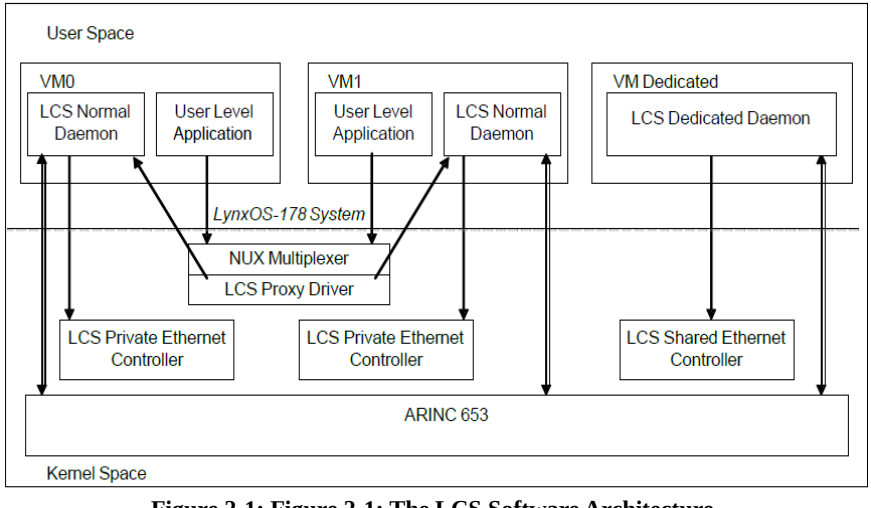

Figure 2-1: Figure 2-1: The LCS Software Architecture

LCS Proxy Socket Driver

The LCS proxy socket driver transmits socket system calls to the LCS networking daemon and passes the replies from the daemon to the user applications.

The LCS proxy socket driver provides the kernel side support of networking sockets. The proxy driver converts socket system calls to the LCS requests and

passes them to the LCS networking daemon. The daemon processes these LCS requests and composes the LCS replies that contain the processed data and returning value of the socket system call. The proxy driver returns information from the LCS reply to the user level application when the daemon puts the LCS reply to the proxy driver device.

The LCS proxy driver has separate queues of the LCS requests/replies for each partition. As the result, one instance of the LCS proxy driver handles all the networking sockets in the system.

LCS NetworkingDaemon

The core functionality of the TCP/IP stack is implemented in the LCS networking daemon. The LCS networking daemon provides the link layer, ARP, IP, ICMP, UDP, and TCP protocols, ability to configure the network interface, and IP packet routing except packet forwarding.

The LCS networking daemon should be launched on every partition that requires the LCS networking facility at the system startup before launching any utility that uses TCP/IP. The daemon utilizes the LCS proxy socket driver to obtain socket requests from applications. Also, the daemon uses the LCS networking controller driver to send and receive raw Ethernet frames to and from the network.

The LCS Networking Daemon Modes

With the LCS TCP/IP stack, it is possible to make the intra-partition Ethernet controller access. The following two modes of the LCS networking daemon are implemented:

- Normal Mode
- Dedicated Mode

These two modes are described in this section.

Normal Mode

In the Normal Mode, the LCS networking daemon provides regular LCS TCP/IP networking functionality in the partitions.

The LCS networking daemon invoked in the Normal Mode owns all private LCS networking controllers that belong to it and interacts directly with these controllers. However, the daemon in this mode does not have a direct access to the shared LCS networking controllers, and receives or sends the Ethernet frames from or to the shared controllers through the LCS networking daemon launched on the other partition in the Dedicated Mode by means of the intra-partition mechanism.

Initially, the daemon reads its configuration file to learn the list of private Ethernet controllers to which it has access and the other system networking configuration parameters.

The daemon opens the LCS proxy socket device and all LCS networking controller devices that belong to it.

The LCS networking daemon opens the intra-partition device, if the daemon intends to use shared LCS networking controller devices.

The LCS networking daemon uses the LCS proxy socket driver to obtain socket requests only from the user level applications. The daemon polls all opened devices and performs the following connective activities:

- When the daemon receives a request (a system call of the user level application) from the LCS proxy socket device, the daemon processes the request and sends prepared data as a reply.
- When the daemon receives the Ethernet frame from the LCS networking controller, the daemon parses the Ethernet frame through the TCP/IP stack protocols.
- When the daemon receives the Ethernet frame from the intra-partition device, the daemon parses the Ethernet frame through the TCP/IP stack protocols.

Dedicated Mode

In the Dedicated Mode, the LCS networking daemon owns all shared LCS networking controllers and sends or receives the Ethernet frames either to or from the network. The LCS networking daemon routes the Ethernet frame between the shared LCS networking controllers and corresponding LCS networking daemons which are launched in Normal Mode. Namely, the dedicated daemon puts the Ethernet frame that is received from the network to the `lcs_ve` device and vice versa.

The dedicated daemon does not have access to the LCS proxy socket drivers as it will gather the requests to the shared controllers from the other partitions and send the replies back.

Initially, the daemon reads the configuration file to learn the lists of shared Ethernet controllers and corresponding partitions.

The LCS networking daemon opens all shared special devices files and intra-partition device. The daemon then polls all opened devices.

- When the Ethernet frame is received from the shared controller, the daemon routes this frame to corresponding normal daemons from other partitions.
- When a frame is read from the intra-partition device, the dedicated daemon sends this frame to the network through corresponding Ethernet controller. The dedicated daemon does not process the Ethernet frames, it distributes the Ethernet frames between partitions.

LCS Networking Controller Drivers

The LCS networking controller drivers provides an interface for the LCS networking daemon to the underlying networking controllers. The drivers transfer Ethernet frames received from the networking controllers to the LCS networking daemon and send raw Ethernet frames back to the network without any processing.

There are two types of the LCS networking controllers:

- Private controllers
- Shared controllers

A private Ethernet controller is used by only one partition (that is, by only one LCS networking daemon). The other partitions do not have access to the device.

An access to the shared Ethernet controllers is handled by means of the dedicated LCS daemon. The LCS networking daemons have access to the shared LCS networking controllers through a dedicated LCS networking daemon, which is the only owner of all shared controllers.

As a result, an instance of the LCS networking controller driver (private or shared) is always handled by only one owner.

The LCS provides support for the following drivers:

- Intel® i350
- Intel® i82571
- Intel® i82572
- Intel®i82576
- Intel®i82580
- QorIQ T2080 MEMAC
- QorIQ T1024 MEMAC
- QorIQT2081 MEMAC

CHAPTER 3 LCS API

LCS provides the POSIX-like socket API which is already present in LynxOS-178. The API functions are described in the LynxOS-178 man pages. The LCS TCP/IP supports the following system calls:

- `socket()`
- `bind()`
- `listen()`
- `accept()`
- `connect()`
- `shutdown()`
- `send()`
- `sendto()`
- `recv()`
- `recvfrom()`
- `setsockopt()`
- `getsockopt()`
- `getsockname()`
- `getpeername`
- `write()read()`
- `select()`
- `close()`
- `fcntl()`
- `ioctl()`

Please refer to the LynxOS-178 man pages for the descriptions of all API functions.

LCS does not support the following system calls that operate with sockets:

- `socketpair()`
- `sendmsg()`

- recvmsg()
- stat()
- lock()
- unlock()

Socket Types

LCS supports the AF_INET POSIX socket address family. The socket types supported in the context of the family are:

- SOCK_STREAM for the TCP protocol
- SOCK_DGRAM for the UDP protocol
- SOCK_RAW for raw sockets.

Also, LCS supports the *SOCK_RAW* socket type of the PF_ROUTE protocol family. However, the LCS TCP/IP stack provides the restricted set of routing messages. Specifically, it provides the following types of routing commands:

- RTM_GET – Report metrics and other route information
- RTM_ADD – Add a route
- RTM_DELETE – Delete a route

Socket Options

The following socket options are supported by LCS:

- IP_TOS
- IP_MULTICAST_IF
- IP_MULTICAST_TTL
- IP_MULTICAST_LOOP
- IP_ADD_MEMBERSHIP
- IP_DROP_MEMBERSHIP

- LCS API
- SO_BROADCAST
- SO_LINGER
- SO_SNDBUF
- SO_RCVBUF
- SO_TYPE
- SO_ERROR
- SO_REUSEADDR
- SO_REUSEPORT
- TCP_NODELAY

Non-blocking Socket Mode

The LCS networking daemon provides a Non-blocking Socket Mode. Regardless of the mode, a thread is blocked and the request is passed to the LCS daemon. If the LCS networking daemon has any data, it will return a reply or return the EAGAIN error.

One socket cannot be handled by several threads simultaneously. If two or more threads make a system call on one socket, only one thread will be served and the other will be blocked. For example, Thread A uses a blocking system call. If it is served first, then Thread B will be blocked until Thread A has been served, even if the Thread B system call is non-blocking.

The Non-blocking Socket Mode is set via the fcntl() system call that provides the O_NONBLOCK descriptor option.

- fcntl(sfd, F_SETFL,FNDELAY) – Set the Non-blocking Mode

ioctl Network Commands

Table 3-1 lists the commands that can be performed on the socket descriptors by the `ioctl` system call.

Table 3-1: ioctl Network Commands

Ioctl Commands	Description
SIOCGIFADDR	Get interface address
SIOCSIFFLAGS	Set interface flags
SIOCGIFFLAGS	Get interface flags
SIOCGIFBRDADDR	Get broadcast address
SIOCGIFCONF	Get interfaces list
SIOCGIFNETMASK	Get network address mask
SIOCSIFNETMASK	Set network address mask
SIOCGIFMETRIC	Get the metric associated with the interface
SIOCSIFMETRIC	Set the metric associated with the interface
SIOCDIFADDR	Delete interface address
SIOCAIFADDR	Add or change the interface alias
SIOCGIFMTU	Get MTU size
SIOCSIFMTU	Set MTU size

CHAPTER 4 *LCS Configuration*

LCS Daemon Configuration

The LCS daemon configuration is provided in the `lcsd.conf` file. For each VM there is a separate file located in the `etc/VM<x>` directory, where `<x>` is a VM number.

Format and Syntax

The LCS daemon configuration file syntax is similar to the syntax of the Virtual Machine configuration file (the `vct` file).

Contents of the File

According to the LCS configuration file syntax, all configuration information is stored either in the table object or in scalar objects.

The LCS configuration file can contain only one table object. The table object is called *section*. The objects that are defined inside the section are called *section objects* and are marked with the section token. Other objects are called *global objects* and are marked with the global token. Global objects cannot be defined inside the section and section objects cannot be defined outside the section.

Scalar objects can be defined either inside the section or outside of it.

Some objects are mandatory and must be specified within the configuration file, the other objects are optional and can be omitted. If an object is omitted, the default value is used.

Table Objects

Section

mandatory, global

The table object is called section. The table object describes a physical interface.

Only one table object can exist in the LCS configuration file.

Section is identified by two tokens: `<lcsX>` and `</lcsX>`, where *X* is a section number. The `<lcsX>` token indicates the beginning of the section and the `</lcsX>` token indicates the ending of the section. All interface parameters specified by *scalar* objects are located between these two tokens. Please note that scalar objects marked as global cannot be specified inside the section object.

Each section defines one physical interface or inter-VM interface. The physical interface must be described in the configuration file to be used by the LCS stack.

Only one VM can use a particular physical or inter-VM interface, but the section number can differ from the physical interface number, so the sections with the same number can exist in different VM's LCS configuration files.

Scalar Objects

UDP_CheckSumReq

not mandatory, global

The value allowed for `UDP_CheckSumReq` is 0 or 1.

If the parameter is 0, the UDP module will not calculate the UDP checksum for the outgoing UDP datagrams.

The default value of this parameter is 1.x

UDP_LargeSendBufferPoolSize

not mandatory, global

The value allowed for this parameter is an integer number from range `[4, NUMTHREADS]` where `NUMTHREADS` is the number of threads running in the VM.

The parameter specifies amount of the IORB pool used to allocate the IORB for large (>MTU) datagrams sending by the user threads. Therefore, this parameter specifies how many threads may send the large datagrams simultaneously without impacting on each other.

The default value of this parameter is 4.

IP_TTLDefVal

not mandatory, global

The value allowed for `IP_TTLDefVal` is a number from the range 0 to 255. This parameter is used as the default TTL value in the IP module.

If the parameter is not specified or has an incorrect value, the default value for TTL is 64.

ARP_DefTimeout

not mandatory, global

The value allowed for `ARP_DefTimeout` is a positive number from the range 0 to 120. This parameter is used as the timer interval value for the old entries flushing in the ARP cache.

If the parameter is not specified, the default value for the ARP module is 3.

Mode

mandatory, global

The `Mode` parameter defines the LCS daemon work mode. The values allowed for this parameter are `Dedicated` or `Normal`. If the parameter is not specified, it will cause an error.

DeviceNodeName

mandatory, section

The `DeviceNodeName` parameter defines a real device name used by the interface.

TCP_SynRetries

not mandatory, global

The `TCP_SynRetries` parameter specifies a number of the SYN retries that will be sent during the TCP active open process.

The value allowed for `TCP_SynRetries` is a number from the range 0 to 25. If the `TCP_SynRetries` parameter is not specified, the default value is 12.

TCP_FirstRto

not mandatory, global

The `TCP_FirstRto` parameter specifies the Retransmission Timeout (RTO) in milliseconds.

The value allowed for `TCP_FirstRto` is a number from the range 0 to 255. If the `TCP_FirstRto` parameter is not specified, the default value is 100.

TCP_MaxRto

not mandatory, global

The `TCP_MaxRto` parameter specifies the maximum RTO value in milliseconds. The value allowed for `TCP_MaxRto` is a number from the range 0 to 10000.

If the `TCP_MaxRto` parameter is not specified, the default value is 1000.

TCP_Msl

not mandatory, global

The `TCP_Msl` parameter specifies the duration of the 2MSL timeout in milliseconds. The value allowed for this parameter is the number from range `0..24*60*60*1000`.

If the `TCP_Msl` is not specified, the default value is 30000 (30 seconds)

TCP_AckOnPush

not mandatory, global

The `TCP_AckOnPush` parameter specifies the behavior of the TCP stack upon receiving the packet with the PUSH flag set. If the parameter is enabled, then the `ACK` packet will be generated immediately. The value allowed for this parameter are either 0 or 1.

If the `TCP_AckOnPush` parameter is not specified, the default value is 0.

This parameter can be turned on if it is necessary to avoid performance problems.

TCP_MaxBufSize

not mandatory, global

The TCP_MaxBufSize option limits the amount of memory allocated for TCP buffers. If the option is missed in the lcsd.conf file, then the default value will be used. The allowed values range from 0 to 64k.

The memory amount specified by this option will be allocated for both TCP send and TCP receive buffers, so to get the total amount allocated for TCP buffers the TCP_MaxBufSize value shall be doubled.

The purpose is to reduce the memory footprint of LCS daemon. The default value is 32k.

With following configuration file the LCS memory footprint is not more than 5M (default memory footprint is about 40M)

```
Mode=Normal;
SockLim=16;
TCP_MaxBufSize=163
84;
<lcs0>
   DeviceNodeName=/dev/ lcs_dtsec0;
</lcs0>
```

IP_FragQueueSize

not mandatory, global

The IP_FragQueueSize parameter specifies the length of the fragmented packet queue. Basically, this is the number of fragmented packets that can be processed simultaneously. Each increment of this parameter consumes 65568 bytes of memory.

The value allowed for IP_FragQueueSize is a number from the range 1 to 2048. The default value is 10.

IP_MaxFragCount

not mandatory, global

The IP_MaxFragCount parameter specifies the maximum number of fragments in each fragmented IP packet. The general rule for calculating this parameter is IP_MaxFragCount = (MAX_IP)/((MIN_MTU - 53)/8*8)+1. All calculations are integer, MIN_MTU is the minimum expected MTU in the path, MAX_IP is the maximum expected size for the original IP packet/datagram. Each increment of this parameter consumes IP_FragQueueSize*2080 bytes of memory.

The value allowed for IP_MaxFragCount is a number from the range 2 to 4093. The default value is 70.

IP_OutFragCount

not mandatory, global

The `IP_OutFragCount` parameter specifies the maximum number of datagrams in each outgoing IP packet what should be fragmented. The value allowed for `IP_OutFragCount` is a number from the range 64 to 4093.

The default value is 64.

SockLim

not mandatory, global

The `SockLim` value limits the global number of sockets the LCS has to support, so it limits the consumption of memory required for supporting the sockets. The allowed values are from 1 to 512. Each socket consumes about 64k of memory.

The purpose is to reduce the memory footprint of LCS daemon. The default value is 256.

Runtime Configuration

This section describes runtime configurations for the Development Mode and Production Mode.

Development Mode

Perform the following steps after the system boots up:

1. Start the LCS networking daemon by executing the following command:

   ```
   [ System: / ] /net/lcsd&
   ```

   Refer to the `lcsd` man page for more information about this command.

2. Configure the networking interfaces by executing the following commands:

   ```
   [ System: / ] ifconfig lcs0 192.168.4.224 up
   [ System: / ] ifconfig lcs1 192.168.5.224 up
   [ System: / ] ifconfig lcs2 192.168.6.224  up
   ```

Please note that all network interfaces should be configured in the `lcsd.conf` file.

Refer to the `ifconfig` man page for more information about this command.

3. (Optional). Configure the routing table by executing the following command:

```
[ System: / ] route add default 192.168.4.254
```

This command adds a default router to `192.168.4.254`.

Refer to the `route` man page for more information about this command.

Production Mode

There is no shell in the Production Mode. It means that all utilities should be started in the user program using `exec` family system calls.

KDI Configuration

Driver Configuration Table

The following entries should be present in the KDI configuration file:

- Proxy device driver

To include the proxy device driver in the kernel, include the lcspx.cfg file in the driver configuration table file config.tbl.

- Interface drivers

The interface drivers are included in the KDI by adding an appropriate entry

(e.g. lcs_dtsec.cfg where dtsec is physical nic of board cw187) in the config.tbl file.

- The inter-VM multiplexer driver (e.g. `lcs_ve.cfg`)
- The LCS daemon file lcsd and other network utilities and configuration files.

NOTE: by default, the lcs "make install" (e.g. `cd $ENV_PREFIX/lcs; make install`) automatically updates `$ENV_PREFIX/bsp.<board>/config.tbl` so it's ready for user to build the KDI.

BSP Configuration

LCS configures a BSP automatically. It is important to understand that old BSP files are backed-up (for every supported BSP) in the following location:

`$(ENV_PREFIX)/ LCS_BSP_backup`

The KDI Specification File

The following is performed automatically by the LCS installation procedure.

It is important to note that the following entries should be present in the KDI specification under the `lynxos-178.spec` file should you wish to modify them:

- The LCS networking daemon executing file (to provide the TCP/IP stack in LCS)

  To include the executing file in the KDI, add the following line to the `directory=/net` section of the `lynxos-178.spec` file:

```
file=lcsd source=$(ENV_PREFIX)/net/lcsd owner=0 group=0 mode=-r-xr-xr-x
```

- The configuration files for each VM on which the LCS networking daemon will be started.

  A VM configuration file `lcsd.conf` is located in `/etc/VM<X>`, where `<X>` is the VM number.

  For example, to add configuration files for three VMs to the KDI, add the following lines to the `directory=/etc` section:

```
directory=/etc/VM0 owner=0 group=0 mode=drwxr-xr-x
file=lcsd.conf source=$(ENV_PREFIX)/etc/VM0/lcsd.<board>.conf \
owner=0 group=0 mode=-rw-r--r--
directory=/etc/VM1 owner=0 group=0 mode=drwxr-xr-x
file=lcsd.conf source=$(ENV_PREFIX)/etc/VM1/lcsd.<board>.conf \
owner=0 group=0 mode=-rw-r--r--
directory=/etc/VM2 owner=0 group=0 mode=drwxr-xr-x
file=lcsd.conf source=$(ENV_PREFIX)/etc/VM2/lcsd.<board>.conf \
owner=0 group=0 mode=-rw-r--r--
```

  Where `<board>` is the cw187

- Network utilities

Add the following lines to the `lynxos-178.spec` file:

- To the `directory=/net` section:

```
file=tftpd source=$(ENV_PREFIX)/net/tftpd owner=0 group=0 mode=-r-xr-
xr-x file=ftpd source=$(ENV_PREFIX)/net/ftpd owner=0 group=0 mode=-r-
xr-xr-x file=inetd source=$(ENV_PREFIX)/net/inetd owner=0 group=0
mode=-r-sr-sr-x
```

- To the `directory=/bin` section:

```
file=arp source=$(ENV_PREFIX)/bin/arp owner=0 group=0 mode=-r-xr-xr-x
file=ifconfig source=$(ENV_PREFIX)/bin/ifconfig \
owner=0 group=0 mode=-r-xr-xr-x
file=ping source=$(ENV_PREFIX)/bin/ping owner=0 group=0 mode=-r-xr-
xr-x file=route source=$(ENV_PREFIX)/bin/route owner=0 group=0 mode=-
r-sr-sr-x file=ftp source=$(ENV_PREFIX)/bin/ftp owner=0 group=0
mode=- r-sr-sr-x file=tftp source=$(ENV_PREFIX)/bin/tftp owner=0
group=0 mode=-r-sr-sr-x file=tcpdump source=$(ENV_PREFIX)/bin/tcpdump
\ owner=0 group=0 mode=-r-sr-sr-x
```

All the entries described above can be omitted if a respective functionality is not required.

Configuring LCS in the Normal Mode

The LCS networking daemon started in the Normal Mode can transmit data either via its own private Ethernet network interfaces or via shared interfaces provided by the other LCS network daemon started in the Dedicated Mode.

This section explains how to configure LCS in the Normal Mode.

Configuring Private Interfaces

The LCS networking daemon can own a private network controller. This controller is accessible only by the LCS networking daemon and cannot be used by other LCS daemons started in other VMs. All interfaces that are required by the VM are described in an LCS networking daemon configuration file.

The LCS networking daemon configuration file is the table object in the LCS configuration file. Refer to “LCS Configuration” on page 19 for the description of the LCS networking daemon configuration file.

Up to 7 interfaces per a VM can be configured in the LCS networking daemon configuration file. The sections describing network interfaces are numbered from 0 to 6 and must go in the ascending order. The loopback interface does not need to be configured in the LCS networking daemon configuration file.

In the example below, the section lcs0 uses the `/dev/lcs_<NIC driver name>` interface. It means that all data passed to the `lcs0` interface go through the device `/dev/lcs_<NIC driver name>`.

```
<lcs0>
    DeviceNodeName=/dev/lcs_dtsec0;
</lcs0>
```

The `DeviceNodeName` parameter specifies a private Ethernet controller driver used in the section. If `DeviceNodeName` is omitted, the `lcs_dtsec` parameter is equal to the section number.

Configuring Interfaces for Inter-Partition Communication

An interface can transmit data either directly to the Ethernet device (that is, to its private interface) or through interfaces owned by the LCS networking daemon started by another VM in the Dedicated Mode. In the latter case, `lcs_ve` is used for inter-partition communication. This section describes how to configure the LCS networking daemon in the Normal Mode to use interfaces shared by the dedicated daemon.

The inter-VM communications are performed using the `lcs_ve` driver. The `lcs_ve` driver acts like an Ethernet switch allowing the broadcast packets to be sent to any VM, except for sender. The unicast packets are sent only to the destination VM. The LCS installation procedure adds `lcs_ve` entries to the config table and dedicates the `lcs6` interface for inter-VM communication purposes. (Note: lcs6 was used as an example, user can use any lcs interface for inter-VM communication)

When VMs are started and have `lcsd` running, `lcs6` can be set to and used for inter-VM communication. Any VM is able to talk with any other VM.

It is possible to have 4 VMs with `lcsd` assigned to each. For example, a 10.0.0. (VMID+1) IP address can be connected to the `lcs6` interface of each VM. VM4 from VM0 uses a VM4 IP address. The dedicated `lcsd` can also be attached to the `lcs_ve`. In this case, the bandwidth of the shared interface can be split between VMs as 100/N; where 100 = 100Mb (considered a standard bandwidth which is supported by the LCS drivers) and N - the number of VMs sharing the interface.

Configuring LCS in Dedicated Mode

The LCS networking daemon started in the Dedicated Mode checks all interfaces and transmits the received data to all interfaces except the interface from which the packet has been received. The functionality of the LCS networking daemon in Dedicated Mode is the same as the functionality of the Ethernet hub.

In the example below, the dedicated daemon is started on VM2 and has two interfaces, one interface for Ethernet and the other interface for the interpartition communication. Both interfaces are specified in the `/etc/VM2/lcsd.conf` file.

The `lcs3` interface is used to communicate via the Ethernet device `/dev/lcs dtsec0`.

```
// /etc/VM2/lcsd.conf file Mode=Dedicated;

<lcs0>
    DeviceNodeName=/dev/lcs_ve0;
</lcs0>

<lcs3>
    DeviceNodeName=/dev/lcs_dtsec0;
</lcs3>
```

Configuration Examples

Single LCS Networking Server in Normal Mode

In the example of this section, the LCS networking daemon starts in the Normal Mode owning only one network interface.

```
// Single LCS Networking Server in Normal
Mode UDP_CheckSumReq=0;
IP_TTLDefVal=10;
Mode=Normal;
ARP_DefTimeout=10;

TCP_SynRetries=10;
TCP_FirstRto=1000;
TCP_MaxRto=1000;

<lcs0>
    DeviceNodeName=/dev/lcs_dtsec0;
</lcs0>
```

One LCS Daemon in Dedicated Mode Two LCS Daemons in Normal Mode There are three LCS networking daemons in the example outlined in this section:

- The dedicated `lcsd` started on the VM2 with two shared and one private interfaces.
- The normal `lcsd` started on the VM1.
- The normal `lcsd` started on the VM0.

The following example shows the configuration file of the dedicated server started on the VM2 with two shared and one private interfaces:

```
// The LCS will be started in the Dedicated Mode owning 2 shared
// interfaces.

Mode=Dedicated;
<lcs0>
    DeviceNodeName=/dev/lcs_ve0;
</lcs0>
<lcs3>
    DeviceNodeName=/dev/lcs_dtsec0;
</lcs3>
```

The following example shows the configuration file of the server started on the VM1 in the Normal Mode:

```
// The LCS Daemon communicates via one shared interface owned by the
// Dedicated LCS Networking Daemon started in another partition.
Mode=Normal;
<lcs0>
    DeviceNodeName=/dev/lcs_ve0;
</lcs0>
```

The following example shows the configuration file of the server started on the VM0 in the Normal Mode with one inter-VM interface and three private interfaces:

```
// The LCS Daemon owns 4 private interfaces and communicates with
// one interface shared by another daemon.

Mode=Normal;
<lcs0>
    DeviceNodeName=/dev/lcs_ve0;
</lcs0>
<lcs1>
    DeviceNodeName=/dev/lcs_dtsec1;
</lcs1>
<lcs2>
    DeviceNodeName=/dev/lcs_dtsec2;
</lcs2>
<lcs3>
```

```
   DeviceNodeName=/dev/lcs_dtsec3;
</lcs3>
```

CHAPTER 5 LCS Daemon Usage

The LCS networking daemon should be started prior to running any program that requires TCP/IP. The LCS daemon configuration parameters are stored in a configuration file. Refer to "LCS Configuration" on page 19 for more information on this file. This Chapter describes additional command line parameters of the LCS networking daemon.

LCS Networking Daemon Command Line Parameters

The LCS networking daemon (`lcsd`) can be started either as a daemon or as a console application. By default, it is started as a console application.

The following options can be used in the `lcsd` command can command line:

- To start `lcsd` as a console application, execute the following command:

  ```
  /net/lcsd
  ```

- To start `lcsd` in the Daemon Mode, execute the following command:

  ```
  /net/lcsd -d
  ```

- To start `lcsd` as a console application in the background, execute the following command:

  ```
  /net/lcsd &
  ```

- By default, the LCS networking daemon reads its configuration from the `/etc/VM<x>/lcsd.conf` configuration file, where `<x>` is the VM number on which `lcsd` is started. To specify a custom location, run the `lcsd` command with the `-f` option. For example, to make `lcsd` read its configuration from the `/etc/my.conf` file, execute the following command:

  ```
  /net/lcsd -f /etc/my.conf
  ```

Starting LCS Networking Daemon

Development Mode

The LCS networking daemon should be started prior to any application that requires TCP/IP functionality. The LCS networking daemon can be started either as a daemon or as a console application as described in “LCS Networking Daemon Command Line Parameters” on page 33 .After the daemon is started, networking interfaces can be configured by executing the `ifconfig` utility, and routes can be added by executing the `route` utility. For example.

```
[ System: / ] /net/lcsd &
[ System: / ] ifconfig lcs0 192.168.4.226 up
[ System: / ] route add default  192.168.4.254
```

Production Mode

Since shell is not available in the Production Mode, there is no difference between starting daemon as a console application or as a daemon. The LCS daemon process should be started from the Production Mode application of each partial partition. Here is the example where `lcsd` is started and one of its interface is configured.

```
int start_app(const char *app, const char *args[]);

int main(void)
{
    const char *lcsd_args[] = {"lcsd", NULL}; c
    onst char *ifconfig_args[] = {"ifconfig",
                                  "lcs0",
                                  "192.168.4.226",
                                  "up",
                                  NULL};
    /* skipped */
    start_app("/net/lcsd", lcsd_args);
    /* give a time for `lcsd' to start */
    sleep(3);
    start_app("/bin/ifconfig",  ifconfig_args);
    /* skipped */
}

int
start_app(const char *app, const char *args[])
{
    int pid;
    pid = fork();
    switch(pid) {
         case 0: /* client - starting application */
            if(execv(app, args) < 0) {
```

```
                perror("execv");
                exit(1);
            }
            break;
        case -1: /* error
            */ perror("fork");
            exit(1);
        default: /* parent */
            return pid;
    }
}
```

APPENDIX A *RFC 1122 Requirements Summary*

This section defines the detailed list of the Network and Transport Layer protocol features and options implemented by LCS.

Table 5-1: UDP

UDP Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
UDP send Port Unreachable	4.1.3.1		X					X
IP Options in UDP								
Pass received IP options to application layer	4.1.3.2	X						X
Application layer can specify IP options in Send	4.1.3.2	X						X
UDP passes IP options down to IP layer	4.1.3.2	X						X
Pass ICMP messages up to application layer	4.1.3.3	X						X
UDP Checksum								
Able to generate/check checksum	4.1.3.4	X					X	
Silently discard bad checksum	4.1.3.4	X					X	

Table 5-1: UDP (Continued)

UDP Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Sender Option to not generate checksum	4.1.3.4			X				X
Default is to checksum	4.1.3.4	X						X
Receiver option to require checksum	4.1.3.4			X				X
UDP Multi-homing								
Pass specific destination address to application	4.1.3.5	X					X	
Application layer can specify Local IP address	4.1.3.5	X					X	
Application layer can specify wild Local IP address	4.1.3.5	X					X	
Application layer notified of local IP address used	4.1.3.5		X				X	
Bad IP source address silently discarded by UDP/IP	4.1.3.6	X					X	
Only send valid IP source address	4.1.3.6	X					X	
UDP Application Interface Services								
Full IP interface of 3.4 for applications	4.1.4	X					X	

Table 5-1: UDP (Continued)

UDP Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Able to specify TTL, TOS, IP options when sending datagram	4.1.4	X						X
Pass received TOS up to application layer	4.1.4			X				X
Push flag								
Aggregate or queue un- pushed data	4.2.2.2			X				X
Sender collapse successive PSH flags	4.2.2.2		X				X	
SEND call can specify PUSH	4.2.2.2			X				X
If cannot: sender buffer indefinitely	4.2.2.2					X		X
If cannot: PSH last segment	4.2.2.2	X					X	
Notify receiving ALP of PSH	4.2.2.2			X				X
Send max size segment when possible	4.2.2.2		X				X	

Table 5-2: TCP

TCP Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Window								
Treat as unsigned number	4.2.2.3	X					X	
Handle as 32-bit number	4.2.2.3		X				X	
Shrink window from right	4.2.2.16				X			X
Robust against shrinking window	4.2.2.16	X					X	
Receiver's window closed indefinitely	4.2.2.17			X			X	
Sender probe zero window	4.2.2.17	X					X	
First probe after RTO	4.2.2.17		X				X	
Exponential backoff	4.2.2.17		X				X	
Allow window stay zero indefinitely	4.2.2.17	X						X
Sender timeout OK connection with zero window	4.2.2.17					X		X
Urgent Data								
Pointer points to last octet	4.2.2.4	X						X
Arbitrary length urgent data sequence	4.2.2.4	X						X
Inform ALP asynchronously of urgent data	4.2.2.4	X						X

Table 5-2: TCP (Continued)

TCP Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Application can learn if/how much urgent data queued	4.2.2.4	X						X
TCP Options								
Receive TCP option in any segment	4.2.2.5	X					X	
Ignore unsupported options	4.2.2.5	X					X	
Cope with illegal option length	4.2.2.5	X					X	
Implement sending and receiving MSS option	4.2.2.6	X					X	
Send MSS option unless 536	4.2.2.6		X					X
Send MSS option always	4.2.2.6			X			X	
Send MSS default is 536	4.2.2.6	X					X	
Calculate effective send segment size	4.2.2.6	X					X	
TCP Checksums								
Sender compute checksum	4.2.2.7	X					X	
Receiver check checksum	4.2.2.7	X					X	
Use clock-driven ISN selection	4.2.2.9	X					X	
Opening connections								

Table 5-2: TCP (Continued)

TCP Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Support simultaneous open attempts	4.2.2.10	X					X	
SYNC-RCVD remembers last state	4.2.2.11	X					X	
Passive Open call interferes with others	4.2.2.18					X		X
Function: simultaneous. LISTENs for same port	4.2.2.18	X					X	
Ask IP for source address for SYNC if necessary	4.2.3.7	X					X	
Otherwise use local address of connection	4.2.3.7	X					X	
OPEN to broadcast/multicast IP address	4.2.3.14					X		X
Silently discard segment to broadcast/multicast address	4.2.2.14	X					X	
Closing connections								
RST can contain data	4.2.2.12		X				X	
Inform application of aborted connection	4.2.2.13	X					X	
Half-duplex close connections	4.2.2.13			X			X	
Send RST to indicate data lost	4.2.2.13		X				X	

Table 5-2: TCP (Continued)

TCP Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
In TIME-WAIT state for 2xMSL seconds	4.2.2.13	X					X	
Accept SYN from TIME- WAIT state	4.2.2.13			X				X
Retransmissions								
Jacobson Slow Start algorithm	4.2.2.15	X					X	
Jacobson Congestion-Avoidance algorithm	4.2.2.15	X					X	
Retransmit with same IP identification	4.2.2.15			X				X
Karn's algorithm	4.2.3.1	X					X	
Jacobson RTO estimation algorithm	4.2.3.1	X					X	
Exponential backoff	4.2.3.1	X					X	
SYN RTO calculation same as data	4.2.3.1		X					X
Recommended initial values and bounds	4.2.3.1		X					X
Generating ACK's:								
Queue out-of-order segments	4.2.2.20		X					X
Process all queued before send ACK	4.2.2.20	X						X

Table 5-2: TCP (Continued)

TCP Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Send ACK for out-of-order segment	4.2.2.21			X			X	
Delayed ACK's	4.2.3.2		X				X	
Delay < 0.5 seconds	4.2.3.2	X					X	
Every 2nd full-size segment ACK'd	4.2.3.2	X					X	
Receiver SWS-Avoidance Algorithm	4.2.3.3	X					X	
Sending data								
Configurable TTL	4.2.2.19	X					X	
Sender SWS-Avoidance Algorithm	4.2.3.4	X					X	
Nagle algorithm	4.2.3.4		X				X	
Application can disable Nagle algorithm	4.2.3.4	X					X	
Connection Failure								
Negative advice to IP on R1 retries	4.2.3.5	X					X	
Close connection on R2 retries	4.2.3.5	X					X	
ALP can set R2	4.2.3.5	X						X
Inform ALP of R1 <= retries < R2	4.2.3.5		X					X
Recommended values for R1, R2	4.2.3.5		X					X

Table 5-2: TCP (Continued)

TCP Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Same mechanism for SYNs	4.2.3.5	X					X	
R2 at least 3 minutes for SYN	4.2.3.5	X						X
Send Keep-alive Packets:	4.2.3.6			X			X	
Application can request	4.2.3.6	X						X
Default is "off	4.2.3.6	X					X	
Only send if idle for interval	4.2.3.6	X					X	
Interval configurable	4.2.3.6	X					X	
Default at least 2 hours	4.2.3.6	X						X
Tolerant of lost ACK's	4.2.3.6	X					X	
IP Options								
Ignore options TCP doesn't understand	4.2.3.8							
Time Stamp support	4.2.3.8	X					X	
Record Rout support	4.2.3.8			X				X
Source Route:				X				X
ALP can specify	4.2.3.8							
Overrides source route in datagram	4.2.3.8	X						X
Build return route from source route	4.2.3.8	X						X
Late source route overrides	4.2.3.8	X						X

Table 5-2: TCP (Continued)

TCP Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
			X					X
Receiving ICMP messages from IP	4.2.3.9							
Destination Unreachable (0,1,5) => inform ALP	4.2.3.9	X						X
Destination Unreachable (0,1,5) => abort connection	4.2.3.9		X					X
Destination Unreachable (2-4) => abort connection	4.2.3.9					X		X
Source Quench => slow start	4.2.3.9		X					X
Time exceeded => tell ALP, don't abort	4.2.3.9		X					X
Parameter problem => tell ALP, don't abort	4.2.3.9		X					X
Address Validation								
Reject OPEN call to invalid IP address	4.2.3.10							
Reject SYN from invalid IP address	4.2.3.10	X					X	
Silently discard SYN to broadcast/multicast address	4.2.3.10	X					X	
		X					X	
TCP/ALPInterface Services								

Table 5-2: TCP (Continued)

TCP Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Error Report mechanism	4.2.4.1							
ALP can disable Error Report Routine	4.2.4.1	X						X
ALP can specify TOS for sending	4.2.4.2		X					X
Passed unchanged to IP	4.2.4.2	X						X
ALP can change TOS during connection	4.2.4.2		X					X
Pass received TOS up to ALP	4.2.4.2		X					X
FLUSH call	4.2.4.3			X				X
Optional local IP address parameter in OPEN	4.2.4.4	X					X	
Implement IP and ICMP	3.1	X					X	
Handle remote multihoming in application layer	3.1	X					X	
Support local multihoming	3.1			X			X	
Meet gateway specs if forward datagrams	3.1	X						X
Configuration switch for embedded gateway	3.1	X						X
Config switch default to non-gateway	3.1	X						X
Auto-config based on number of interfaces	3.1					X		X

Table 5-2: TCP (Continued)

TCP Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Able to log discarded datagrams	3.1		X				X	
Record in counter	3.1		X				X	
Silently discard Version != 4	3.2.1.1	X					X	
Verify IP checksum, silently discard bad datagram	3.2.1.2	X					X	
Addressing:								
Subnet addressing (RFC- 950)	3.2.1.3	X					X	
Src address must be host's own IP address	3.2.1.3	X					X	
Silently discard datagram with bad destination address	3.2.1.3	X					X	
Silently discard datagram with bad source address	3.2.1.3	X					X	
Support reassembly	3.2.1.4	X					X	
Retain same Id field in identical datagram	3.2.1.5			X				X
TOS:								
Allow transport layer to set TOS	3.2.1.6	X						X
Pass received TOS up to transport layer	3.2.1.6		X					X

Table 5-2: TCP (Continued)

TCP Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Use RFC-795 link-layer mappings for TOS	3.2.1.6				X			X
TTL								
Send packet with TTL of 0	3.2.1.7					X		X
Discard received packets with TTL < 2	3.2.1.7					X		X
Allow transport layer to set TTL	3.2.1.7	X						X
Fixed TTL is configurable	3.2.1.7	X					X	
IP Options:								
Allow transport layer to send IP options	3.2.1.8	X						X
Pass all IP options rcvd to higher layer	3.2.1.8	X						X

Table 5-3: Internet Layer

Internet Layer Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
IP layer silently ignore unknown options	3.2.1.8	X					X	
Security option	3.2.1.8a			X				X

Table 5-3: Internet Layer (Continued)

Internet Layer Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Send Stream Identifier option	3.2.1.8b				X			X
Silently ignore Stream Identifier option	3.2.1.8b	X					X	
Record Route option	3.2.1.8d			X				X
Timestamp option	3.2.1.8e			X				X
Source Route Option:								
Originate & Terminate Source Route options	3.2.1.8c	X						X
Datagram with completed SR passed up to TL	3.2.1.8c	X						X
Build correct (non-redundant) return route	3.2.1.8c	X						X
Send multiple SR options in one header	3.2.1.8c					X		X
ICMP:								
Silently discard ICMP msg with unknown type	3.2.2	X					X	
Include more than 8 octets of original datagram	3.2.2c			X				X
Included octets same as received	3.2.2	X						X
Demux ICMP Error to transport protocol	3.2.2	X						X
Send ICMP error message with TOS=0	3.2.2		X					X

Table 5-3: Internet Layer (Continued)

Internet Layer Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Send ICMP error message for:								
- ICMP error msg	3.2.2					X		X
- IP broadcast or IP multicast	3.2.2					X		X
- Link-layer broadcast	3.2.2					X		X
- Non-initial fragment	3.2.2					X		X
- Datagram with non-unique source address	3.2.2					X		X
Return ICMP error msgs (when not prohibited)	3.3.8	X						X
Dest Unreachable:								
Generate Dest Unreachable (code 2/3)	3.2.2.1		X					X
Pass ICMP Dest Unreachable to higher layer	3.2.2.1	X						X
Higher layer act on Dest Unreach	3.2.2.1		X					X
Interpret Dest Unreach as only hint	3.2.2.1	X						X
Redirect:								
Host send Redirect	3.2.2.2				X			X
Update route cache when recv Redirect	3.2.2.2	X						X
Handle both Host and Net Redirects	3.2.2.2	X						X

Table 5-3: Internet Layer (Continued)

Internet Layer Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Discard illegal Redirect	3.2.2.2		X				X	
Source Quench:								
Send Source Quench if buffering exceeded	3.2.2.3			X				X
Pass Source Quench to higher layer	3.2.2.3	X						X
Higher layer act on Source Quench	3.2.2.3		X					X
Time Exceeded: pass to higher layer	3.2.2.4	X						X
Parameter Problem:								
Send Parameter Problem messages	3.2.2.5		X					X
Pass Parameter Problem to higher layer	3.2.2.5	X						X
Report Parameter Problem to user	3.2.2.5			X				X
ICMP Echo Request or Reply:								
Echo server and Echo client	3.2.2.6	X					X	
Echo client	3.2.2.6		X				X	
Discard Echo Request to broadcast address	3.2.2.6			X			X	
Discard Echo Request to multicast address	3.2.2.6			X			X	
Use specific-dest addr as Echo Reply src	3.2.2.6	X					X	

Table 5-3: Internet Layer (Continued)

Internet Layer Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Send same data in Echo Reply	3.2.2.6	X					X	
Pass Echo Reply to higher layer	3.2.2.6	X						X
Reflect Record Route, Time Stamp options	3.2.2.6		X					X
Reverse and reflect Source Route option	3.2.2.6	X						X
ICMP Information Request or Reply:	3.2.2.7				X			X
ICMP Timestamp and Timestamp Reply:	3.2.2.8			X				X
Minimize delay variability	3.2.2.8		X					X
Silently discard broadcast Timestamp	3.2.2.8			X				X
Silently discard multicast Timestamp	3.2.2.8			X				X
Use specific-dest addr as TS Reply src	3.2.2.8	X						X
Reflect Record Route, Time Stamp options	3.2.2.8		X					X
Reverse and reflect Source Route option	3.2.2.8	X						X
Pass Timestamp Reply to higher layer	3.2.2.8	X						X
Obey rules for "standard value	"3.2.2.8	X						X

Table 5-3: Internet Layer (Continued)

Internet Layer Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
ICMP Address Mask Request and Reply:								
Addr Mask source configurable	3.2.2.9	X						X
Support static configuration of addr mask	3.2.2.9	X					X	
Get addr mask dynamically during booting	3.2.2.9			X				X
Get addr via ICMP Addr Mask Request/Reply	3.2.2.9			X				X
Retransmit Addr Mask Req if no Reply	3.2.2.9	X						X
Assume default mask if no Reply	3.2.2.9		X					X
Update address mask from first Reply only	3.2.2.9	X						X
Reasonableness check on Addr Mask	3.2.2.9		X					X
Send unauthorized Addr Mask Reply msgs	3.2.2.9					X		X
Explicitly configured to be agent	3.2.2.9	X						X
Static config=> Addr-Mask-Authoritative flag	3.2.2.9		X					X
Broadcast Addr Mask Reply when init.	3.2.2.9	X						X

Table 5-3: Internet Layer (Continued)

Internet Layer Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
ROUTING OUTBOUND DATAGRAMS:								
Use address mask in local/remote decision	3.3.1.1	X					X	
Operate with no gateways on conn network	3.3.1.1	X					X	
Maintain "route cache" of next-hop gateways	3.3.1.2	X					X	
Treat Host and Net Redirect the same	3.3.1.2		X					X
If no cache entry, use default gateway	3.3.1.2	X					X	
Support multiple default gateways	3.3.1.2	X						X
Provide table of static routes	3.3.1.2			X			X	
Flag: route override by Redirects	3.3.1.2			X				X
Key route cache on host, not net address	3.3.1.3			X			X	
Include TOS in route cache	3.3.1.3		X					X
Able to detect failure of next-hop gateway	3.3.1.4	X						X
Assume route is good forever	3.3.1.4					X	X	
Ping gateways continuously	3.3.1.4					X		X

Table 5-3: Internet Layer (Continued)

Internet Layer Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Ping only when traffic being sent	3.3.1.4	X						X
Ping only when no positive indication	3.3.1.4	X						X
Higher and lower layers give advice	3.3.1.4		X					X
Switch from failed default gateway to another	3.3.1.5	X						X
Manual method of entering config info	3.3.1.6	X					X	
REASSEMBLY and FRAGMENTATION:								
Able to reassemble incoming datagrams	3.3.2	X					X	
At least 576 byte datagrams	3.3.2	X					X	
EMTU_R configurable or indefinite	3.3.2		X					X
Transport layer able to learn MMS_R	3.3.2	X					X	
Send ICMP Time Exceeded on reassembly timeout	3.3.2	X						X
Fixed reassembly timeout value	3.3.2		X					X
Pass MMS_S to higher layers	3.3.3	X					X	

Table 5-3: Internet Layer (Continued)

Internet Layer Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Local fragmentation of outgoing packets	3.3.3			X			X	
Else don't send bigger than MMS_S	3.3.3	X					X	
Send max 576 to off-net destination	3.3.3		X				X	
All-Subnets-MTU configuration flag	3.3.3			X				X
MULTIHOMING:								
Reply with same addr as spec-dest addr	3.3.4.2		X				X	
Allow application to choose local IP addr	3.3.4.2	X					X	
Silently discard datagram in "wrong" interface	3.3.4.2			X			X	
Only send datagram through "right" interface	3.3.4.2			X				X
SOURCE_ROUTE FORWARDING:								
Forward datagram with Source Route option	3.3.5			X				X
Obey corresponding gateway rules	3.3.5	X						X
Update TTL by gateway rules	3.3.5	X						X
Able to generate ICMP err code 4, 5	3.3.5	X						X

Table 5-3: Internet Layer (Continued)

Internet Layer Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
IP src addr not local host	3.3.5			X				X
Update Timestamp, Record Route options	3.3.5	X						X
Configurable switch for non-local SRing	3.3.5	X						X
Defaults to OFF	3.3.5	X						X
Satisfy gateway access rules for non-local SRing	3.3.5	X						X
If not forward, send Dest Unreach (cd 5)	3.3.5		X					X
BROADCAST:								
Broadcast addr as IP source addr	3.2.1.3					X		X
Receive 0 or -1 broadcast format OK	3.3.6		X				X	
Configurable option to send 0 or -1 broadcast	3.3.6			X				X
Defaults to -1 broadcast	3.3.6		X				X	
Recognize all broadcast address formats	3.3.6	X					X	
Use IP broadcast/multicast address in link-layer broadcast	3.3.6	X					X	
Silently discard link-layer- only broadcast	3.3.6		X				X	

Table 5-3: Internet Layer (Continued)

Internet Layer Requirements Summary	RFC-1122 Section	Exact RFC-1122 Specification					Available In LynxOS-178 v2.2.4	
		Must	Should	May	Shout Not	Must Not	Is	Is Not
Use Limited Broadcast addr for connected net	3.3.6		X				X	
MULTICAST:								
Support local IP multicasting (RFC-1112)	3.3.7		X				X	
Support IGMP (RFC-1112)	3.3.7			X				X
Join all-host group at startup	3.3.7		X					X
Higher layers learn interface multicast capability	3.3.7		X					X
INTERFACE:								
Allow transport layer to use all IP mechanisms	3.4	X						X
Pass interface up to transport layer	3.4	X						X
Pass all IP options up to transport layer	3.4	X						X
Transport layer can send certain ICMP messages	3.4	X						X
Pass specified ICMP messages up to transport layer	3.4	X						X
Include IP header+8 octets or more from orgin	3.4	X						X
Able to leap tall buildings at a single bound	3.5		X				X	

APPENDIX B *LynxOS-178 Network Utilities and LCS Compatible Features*

Packages and Utilities Providing Application Layer Support

The following utilities are compatible with the LCS TCP/IP stack in addition to the network and transportlayers:

- The `ftp` and `ftpd` utilities to support FTP
- The `tftp` and `tftpd` utilities to support TFTP
- The `ntpd` utility to support SNTP
- The `net-snmp-5.1.1` utility to support SNMPv3

The remote operation and command execution are supported by the `rsh`, `rlogin`, `rcp`, and `rexec`utilities.

Auxiliary LCS Utilities and Libraries

Additionally, LCS is compatible with the following auxiliary utilities and libraries that provide runtime configuration and analyzing facilities:

- `ifconfig`
- `route`
- `arp`
- `ping`
- `tcpdump`
- `libpcap`
- `lcsnetstat`