DUAL – A Packet Format for a New Amateur Radio Link Layer [1]

Omid Mohammadi
Warren Toomey VK1XWT
Lawrie Brown VK1KLB

Department of Computer Science
University College
The University of New South Wales
Australian Defence Force Academy
Canberra, ACT
February 3, 1995

Technical Report CS01/95

Abstract

The standard amateur packet radio protocol, AX.25, provides the end-user with a virtual circuit interface to the link layer, while implementing this using packets, and using a CSMA medium access control mechanism. This paper suggests that the link layer in this application area should be separated into 3 layers (LLC, MAC and Physical Layer), and a set of modular protocols for each part be designed for amateur radio use. A new simplified Logical Link Control layer, DUAL, is described, which is not only more suitable than AX.25 for the transmission of higher-layer traffic, but also more bandwidth-efficient. This is achieved by using link layer addresses tailored to the network protocol used, minimizing the link layer overheads, and by incorporating a modified IP header compression scheme adapted for use in a multi-point environment.

Omid Mohammadi,	Internet e-mail:	omidm@cs.adfa.oz.au
Warren Toomey,	Internet e-mail:	wkt@cs.adfa.oz.au
	AMPRnet e-mail:	vk1xwt@minnie.vk1xwt.ampr.org
	BBS e-mail:	vk1xwt@vk1bbs.act.aus.oc
Lawrie Brown,	Internet e-mail:	Lawrie.Brown@adfa.oz.au
	AMPRnet e-mail:	vk1klb@joruth.vk1klb.ampr.org
	BBS e-mail:	vk1klb@vk1bbs.act.aus.oc

[1] This research was supported in part by ATERB grant N033/011 1992/1993

Contents

1 Introduction 1

2 Separation Of Link Layer Functions 1

3 Previous Protocols – AX.25 2
- 3.1 Frame Structure . . . 3
- 3.2 Bit Stuffing . . . 4
- 3.3 Packet Radio Media Access Protocols . . . 4
- 3.4 TNC – Terminal Node Controller . . . 4
- 3.5 AX.25 – Why Not Suitable . . . 5

4 DUAL – An Unreliable Amateur Link Format 5

5 AX.25 vs. DUAL 6

6 The DUAL Packet Format 7
- 6.1 Protocol Identification . . . 7
- 6.2 Address Type . . . 8
- 6.3 Source And Destination Addresses . . . 8
- 6.4 Data . . . 8
- 6.5 Checksum . . . 8
- 6.6 Comments . . . 8

7 Protocol Ids 8

8 Transporting IP Over DUAL – PR_IP 9
- 8.1 Addresses For IP . . . 9
- 8.2 Broadcast Addresses . . . 10
- 8.3 The CRC Field . . . 10
- 8.4 Limitations And Disadvantages Of TCP/IP Over DUAL . . . 10
- 8.5 Example IP Packets Over DUAL . . . 10

9 Transporting Compressed IP Over DUAL – PR_CIP 11
- 9.1 Protocol Identification . . . 11
- 9.2 Addresses . . . 11

10 Broadcasting With DUAL 11
- 10.1 Broadcasting Link Addresses – PR_BCAST/AD_CALL . . . 12
  - 10.1.1 AD_CALL . . . 12
- 10.2 Broadcasting Textual Information – PR_BCAST/AD_BEACON . . . 13
  - 10.2.1 AD_BEACON . . . 13
- 10.3 Callsign Policy . . . 13

11 DUAL Implementation 13
- 11.1 Implementation Overview . . . 14
- 11.2 Implementing DUAL In The 4.x BSD Network Stack . . . 14
- 11.3 The Compression Algorithm . . . 15
  - 11.3.1 Why Compression Is Desirable . . . 15
  - 11.3.2 The Basic Idea . . . 15
  - 11.3.3 The Ugly Details . . . 16
  - 11.3.4 Compressed Packet Format . . . 17
- 11.4 Modifications To The Compression Algorithm . . . 18

12 Conclusion 18

13 Further Work 18

A The KISS TNC: A Simple Communications Protocol **19**
A.1 Asynchronous Frame Format 19
A.2 Control Of The KISS TNC 20
A.3 Using DUAL With KISS Modems 20

B Installation Guide **21**
B.1 Introduction 21
B.2 Installing The Code 21
B.3 Building The Code 21
B.4 Rebuilding The Kernel 21

Bibliography **22**

1 Introduction

The current Amateur Link Protocol, AX.25 [WB4JFI 84], has many deficiencies. This paper proposes a new Link Protocol called DUAL which attempts to address these deficiencies. DUAL - An Unreliable Amateur Link Format is *not* a replacement for AX.25 – it does not have all of AX.25's functionality. However, DUAL is designed to be a flexible method for transporting higher-level network packets. If required, AX.25 functionality can be added to DUAL as a Network layer protocol.

DUAL attempts to:

- Allow the transportation of packets from many types of higher-layer protocols, using link addresses that are suitable for these protocols instead of the standard amateur callsign,
- Provide a mechanism to broadcast the association between these link addresses and the callsign of the transmitting operator,
- Minimise the overhead of the Link layer, in terms of packet header size,
- Separate the Logical Link Control, Medium Access Control and Physical layers in a modular way, so that DUAL can be used independently of the Physical, MAC and higher layers desired.

The main reason behind the design of DUAL was to create a link layer that was better suited for the transmission of TCP/IP packets across amateur radio [Kahn 77] than the existing protocol. Although the design has been generalised to allow many other protocols, a large proportion of this document covers TCP/IP transmission over DUAL.

2 Separation Of Link Layer Functions

The data communication system must provide each workstation/computer on the link with the capability to send data to another station, and the sending station must be assured that the data arrive error-free at the receiving station. The sending and receiving stations must maintain complete accountability for all traffic. In the event the data are distorted, the receiver site must be able to direct the originator to retransmit the data.

The IEEE Committee has developed a three-layer architecture for local network access as illustrated in Figure 1.

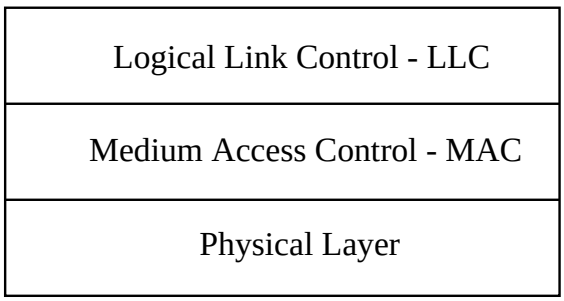

Figure 1: IEEE Reference Model

The IEEE Committee split the data link layer into two sublayers: Media Access Control (MAC) and Logical Link Layer (LLC). This sublayer was implemented to make the LLC sublayer independent of a specific LAN access method. The LLC also is used to provide an interface into or out of the specific MAC protocol.

LLC provides the interface to the next upper-layer protocol, which can be a user application, but more often is a network layer protocol such as the widely used Internet Protocol (IP). As part of these interfaces, LLC is responsible for:

- On transmission, assemble data into a frame with address and CRC fields.
- On reception, disassemble frame, perform address recognition and CRC validation.

The MAC sublayer consists of other sublayers. These sublayers are organized as transmit or receive entities. And finally, at the lowest layer, are the functions generally associated with the physical layer. For more detailed discussion about Link Control Protocols see [Stallings 91] and [Black 93]

From the discussion above it is clear that the MAC/LLC split provides several attractive features:

- It Controls access to the shared channel among the autonomous user devices.
- It provides for a decentralized (peer-to-peer) scheme that reduces the LAN's susceptibility to errors.
- It provides a more compatible interface with wide area networks, since LLC is a subset of HDLC [Black 93].
- LLC is independent of a specific access method; MAC (not LLC) is protocol-specific.

This approach gives an IEEE LAN architecture a flexible interface with workstations and other networks. It also make the LLC sublayer independent of a specific LAN access method. Note also that a set of modular protocols for each part has the advantage of hiding the details from one sublayer to another, and also makes it easier to understand.

3 Previous Protocols – AX.25

Packet radio is a mode of communications that will link Amateur Radio stations together directly or by means of a network. In contrast to the case with the packet-switched networks, the key issue is transmission capacity allocation, and has the following characteristics:

- Packet transmission.
- No switching.
- Reception by many or all stations.
- Common transmission medium.
- Automatic routing throughout a global data network.

An effort to standardize these networks has been underway since 1982, under the sponsorship of the American Radio Relay League [Karn *et al* 85] and [Diersiong & Ward 89], which has produced a standard for a link level protocol suitable for packet-radio networks known as AX.25 [WB4JFI 84].

An ARRL-type network is a distributed network, organized into clusters of stations connected by repeaters. All stations and repeaters share a single frequency for transmission and reception. The AX.25 standard does not specify the frequency to be used, but a typical network uses the 220-MHz band, using FSK and with a bandwidth of 20 kHz or 100 kHz. A typical data rate is 4800 bps, and a fixed routing scheme is used. In this case, the route to be followed is specified by the source station.

AX.25 Link-Layer Protocol is designed to be used between two Amateur Radio stations in a point-to-point communications environment. The Link Protocol is based on, and very close to, HDLC [Black 93]. The following subsections explains the AX.25 frame structure very briefly.

3.1 Frame Structure

Link layer packet radio transmissions are send in small blocks of data, called *frames*. Each frame is made up of several smaller groups, called *fields*.

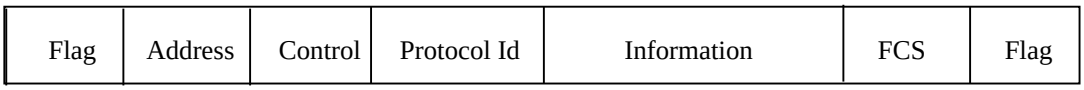

Figure 2: AX.25 Frame Format

Figure 2 depicts the structure of the AX.25 frame. The frame format is as follows:

- *Flag (1 octet)*: as in HDLC, the flag field is used to delimit frames, it occurs at both the beginning and end of each frame, and has the value of 7E hex or 01111110. This sequence is not allowed to occur anywhere else inside a complete frame.
- *Address (14 to 70 octets)*: explained below.
- *Control (1 octet)*: as in HDLC, the control field is used to identify the type of frame being passed and control several attributes of the level 2 connection.
- *Protocol Identifier (1 octet)*: specifies what kind of network layer (layer 3) protocol, if any, is in use.
- *Information (up to 256 octets)*: as in HDLC, the information field is used to convey user data from one end of the link to the other. Any information in the I field shall be passed along the link transparently, except for the zero-bit insertion (see below) necessary to prevent flags from accidentally appearing in the Information field.
- *Frame Check Sequence (2 octets)*: as in HDLC, the frame check sequence is calculated by both the sender and receiver of a frame. It is used to insure that the frame was not corrupted by the medium used to get the frame from the sender to the receiver. It shall be calculated in accordance with ISO 3390 Recommendations.
- *Flag (1 octet)*: as in HDLC.

There are two differences between the AX.25 frame format and that of HDLC: the Protocol Identifier (PID) field and the address field. The PID field is used to designate the layer 3 protocol that is using the AX.25 link protocol. This would allow multiple users of the link layer protocol. The most important difference between AX.25 and HDLC is the addressing technique. In HDLC, there two possible configurations: a point-to-point link with two stations, and a multi-drop link with one primary and multiple secondaries. In either case, a single address is not sufficient in a packet-radio network for two reasons:

1. Since the network is a peer, distributed network, both the source and destination stations should be identified; neither is unique. For flow control, error control, and sequence numbering, both addresses are needed.
2. If repeaters are involved, these repeaters must be specified. In particular, it is the responsibility of the transmitting station to specify the repeater or repeaters that must be used to get from source to destination.

The AX.25 address field is from 14 to 70 octets long, depending on whether and how many repeaters are used between a particular source-destination pair. If the sending and receiving stations are in the same cluster (within range of each other), then it is only necessary to specify the source and destination station addresses. Each is specified using 7 octets, which contains a callsign of up to 7 characters. If a frame is to go through a repeater, an additional address subfield is appended to the end of the address field. For more comprehensive discussion about how to encode the AX.25 address field see [WB4JFI 84].

3.2 Bit Stuffing

In order to assure that arbitrary bit patterns can be inserted into the data field of the frame, the sending station shall monitor the bit sequence for a group of five or more contiguous one bits. Any time five contiguous one bits are sent the sending station shall insert a zero bit after the fifth one bit. During frame reception, any time five contiguous one bits are received, a zero bit immediately following five one bits shall be removed.

3.3 Packet Radio Media Access Protocols

A medium access control protocol is needed for a multi-access network. Such protocols are designed to address the problem of how to share a common broadcast channel. The techniques that have been used for packet radio networks can be termed *random access* or *contention* techniques. They are random access in the sense that there is no predictable or scheduled time for any station to transmit: Stations generate packets for transmission at random times. They are contention in the sense that no central control is exercised to determine whose turn it is. All stations contend for time on a network.

There are three medium access protocol appropriate for packet radio networks: *ALOHA*, *slotted ALOHA*, and *Carrier Sense Multiple Access* (CSMA). This subsection introduce CSMA very briefly, for more comprehensive discussion about these topics see [Stallings 91].

The maximum channel utilization is only 0.37 with slotted ALOHA (see [Stallings 91]). The reason for this poor performance is that both ALOHA and slotted ALOHA fail to take advantage of one of the key properties of packet radio networks, which is that *the propagation delay between stations is insignificant compared to packet transmission time*.

If the station-to-station propagation time is extremely small compared to packet transmission time, then, when a station launches a packet, all the other stations know it almost immediately. So, if they had any sense, they would not try transmitting until the first station was done. In that case, collision would be rare since they would occur only if two stations began to transmit almost simultaneously. In other word the short delay time provides the stations with better feedback about the state of the system.

This observation led to the development of a technique known as *carrier sense multiple access* (CSMA) or *listen before talk* (LBT). A station wishing to transmit first listens to the medium to determine if another transmission is in progress. If the medium is in use, the station backs off some period of time and tries again. If the medium is idle, the station may transmit. To avoid collisions, a station waits a reasonable amount of time after transmitting for an acknowledgment, taking into account the maximum round-trip propagation delay. If there is no acknowledgment, the station assumes that a collision has occurred and retransmits.

With CSMA, an algorithm is needed to specify what a station should do if the medium is found to be busy. The following three algorithms are in use (see [Stallings 91]):

- *non-persistent* CSMA.
- *1-persistent* CSMA.
- *p-persistent* CSMA.

Both non-persistent and p-persistent algorithms are built into most TNCs, but the default algorithm is non-persistent because it is simpler to use.

3.4 TNC – Terminal Node Controller

Amateur radio packet data communications has been done primarily with dedicated microprocessor systems called *Terminal Node Controllers* (TNCs). The TNC is typically connected to a host computer or terminal with an RS-232 DCE (Data Communications Equipment) connector. This allows any host with asynchronous serial communications capability (e.g. a personal computer or a mainframe) to use the TNC as if it were a conventional telephone mode. A TNC handles all necessary channel protocols, and since

they are often difficult to implement on personal computers which do not provide real-time multitasking operating systems, this approach allows the use of packet radio in a much wider variety of applications.

A TNC is generally used to implement the chosen media access protocols (explained in previous section) in an amateur packet radio network. It may also implement higher layers protocols such as AX.25 and a simple PMS (Personal Messaging System), or running in KISS mode (see appendix A) may off-load these functions to another system.

3.5 AX.25 – Why Not Suitable

As stated earlier the key issue in the packet radio network is transmission capacity, and from the discussion above it is clear that AX.25 doesn't use this capacity efficiently because:

- The virtual circuit functionality is strictly not the responsibility of the link layer. It should form part of the network layer which is separated from the link layer.
- Similarly, source routing is strictly not the responsibility of the link layer. It should form part of the network layer which is separated from the link layer.
- Callsigns are transmitted in every packet, although most amateur services require a transmitter to broadcast their callsign at larger intervals (e.g 10 minutes). As each callsign is 7 octets long, and there are at least two in each packet, this forms a large overhead.

4 DUAL – An Unreliable Amateur Link Format

DUAL, the Digital Unreliable Amateur Link format, provides an amateur link layer with:

- a simple logical link control mechanism for the unreliable transmission and reception of link layer packets, and
- a standard but flexible format for these packets.

DUAL sits as the basic Logical Link Control layer in the overall Link Layer, as shown in the following diagram.

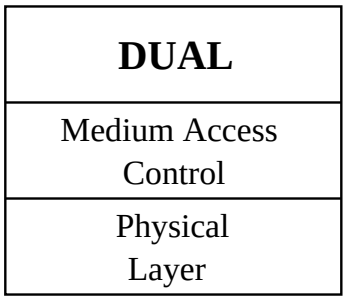

Figure 3: DUAL within the Link Layer

Below DUAL is the Medium Access Layer, which controls access to the link and performs other tasks such as bit/byte-stuffing. At the bottom is the Physical Layer, which transmits the data over the medium using some form of modulation, and perhaps with some data encoding.

DUAL is designed to accept packets from higher layers, and to pass them on to the appropriate MAC and Physical layers for encoding and eventual transmission. Similarly, it accepts packets from MAC layers and passes them on to the appropriate higher layer. Although DUAL will most often be used with a single higher layer and MAC layer, the following scenario is quite possible.

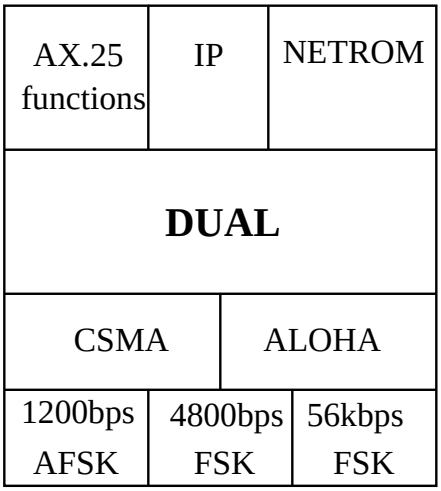

Figure 4: An Example Protocol Stack with DUAL

Here DUAL has access to two MAC layers, a CSMA-style one like that used in AX.25, and ALOHA. Below that are three physical interfaces which can transmit data. Above DUAL are three Network layers, IP, NETROM and a layer which provides AX.25-style virtual circuit connections[2].

5 AX.25 vs. DUAL

AX.25 [WB4JFI 84], the current Amateur Link Protocol, has many deficiencies, which DUAL attempts to address. DUAL is designed to accept packets from higher layers, and to pass them on to the appropriate MAC and Physical layers for encoding and eventual transmission. The receiving entity takes no remedial actions; if the frame is in error, due to transmission noise, it is discarded, and the receiver is not informed about the loss of the traffic. Error recovery is a function of upper-layer protocols, like TCP.

DUAL is configured to support connectionless-mode operations. This mode dose not set up a logical link connection; instead it manages the link layer frames as independent and separate entities. No relationship is maintained between successive data transfer, and moreover, ACKs and NACKs are not provided. Error checking is performed on the frame by the Checksum field.

Since DUAL is connectionless, there will be no requests for retransmissions of bad frames, and collision may also occur, with the potential of losing the frames that collided[3].

Because of the connectionless nature of DUAL the following procedures used in the AX.25 protocol are not necessary:

- Frame sequence number
- Frame rejection and retransmission procedure
- P/F bit procedure
- Link set up and disconnection procedure
- Collision recovery
- Command/response procedure

These are part of the reliable virtual circuit capabilities found in AX.25, which have been removed to minimise the header size in DUAL. The address field operation is different in DUAL which has been explained in section 6.

[2] Neither AX.25 nor Netrom-style functionality are yet designed, and we have no plans to design this functionality at this stage.

[3] Collision detection and recovery is the responsibility of the MAC layer below DUAL.

6 The DUAL Packet Format

DUAL's packet format is designed to be minimal, flexible, and to meet the needs of the both underlying MAC and Physical layers and the Network layers above DUAL.

In order to minimise header sizes, we have removed the reliable virtual circuit capabilities found in AX.25. This dramatically decreases the size of DUAL's headers. At the same time we enabled DUAL to tailor its Link layer addresses so they were more useful to the Network layers above DUAL. This again helps to reduce the size of DUAL's headers.

Note in particular that there is no field to describe the size of the Data field. We believe this is unnecessary. Both the MAC layers and higher layers present DUAL with a packet of a fixed number of octets. Therefore DUAL can determine where the Checksum field is located, and hence the size of the Data field.

The packet format is shown in figure 5:

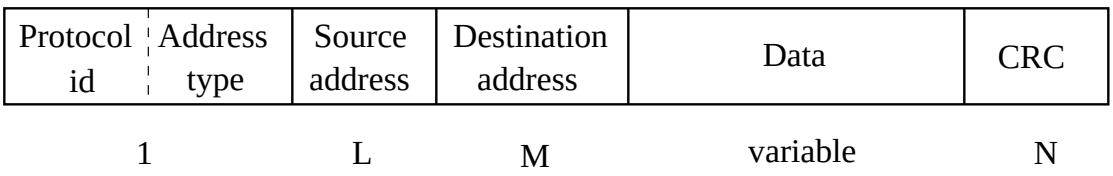

Figure 5: DUAL's Packet Format

DUAL encapsulates some higher layer data in a packet which contains the link addresses of both the source and destination, a higher layer protocol field, and a checksum.

The protocol field is divided into a Protocol Id and an Address Type, allowing $2^5 = 32$ higher layer protocols, and $2^3 = 8$ packet formats per protocol.

The rationale behind permitting eight types of link addresses for each higher level protocol is to allow the link addresses to be tailored to the characteristics of the physical link (speed, quality) and to the addressing needs of the link (if there are only 100 stations on the link, a link address of 1-2 octets is sufficient). Section 8 gives as an example the link addressing scheme used when IP packets are transported over DUAL.

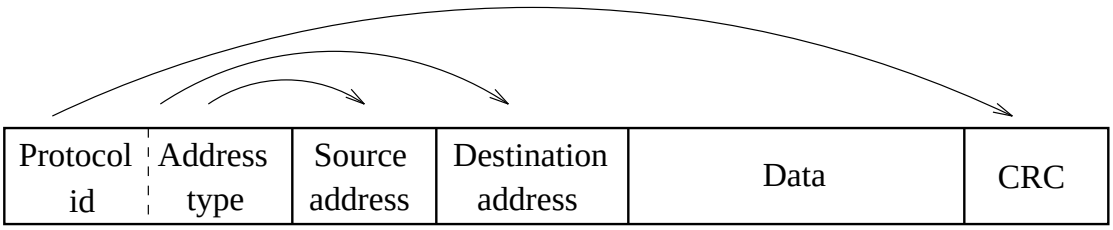

Figure 6: DUAL Field Dependencies

The size and format/content of the Source, Destination and Checksum fields are determined entirely by the Protocol-id/Address-type field. This allows the size and format of these fields to be tailored to suit the protocol being transmitted over the link.

6.1 Protocol Identification

The Protocol Id field is unsigned and 5 bits wide. This indicates to which upper level protocol the data in the packet belongs. The currently defined values for this field are given in Section 7.

5 bits allows DUAL to support up to 32 higher level protocols. We believe this to be sufficient for the near future. An implementation of DUAL is not expected to know about all 32 protocols; it must, however, know at least one.

6.2 Address Type

The Address Type field is unsigned and 3 bits wide. Combined with the Protocol Id field, this field indicates the size and format of the Source and Destination link addresses. This allows each upper level protocol to have eight types of link address types when operating with DUAL.

6.3 Source And Destination Addresses

These fields contain the link address of the node which transmits the packet and node which is the destination of the packet, respectively. The format of each field is determined by both the Protocol Id and Address Type fields in the packet. Note that the two addresses need not be the same size or format; this provides more flexibility for choosing link addresses by the higher layer protocol. However, both must be an integral number of octets wide.

6.4 Data

The Data field contains the data payload of the packet. The format depends on the upper level protocol described by the Protocol id.

6.5 Checksum

This unsigned variable typically contains a checksum over the entire packet, i.e all the fields described above. However, the actual size and contents of this field is determined again by both the Protocol Id and Address Type fields in the packet. This allows the use of no checksums, or checksums that are appropriate for the type of lower layers, e.g a forward error correcting checksum when low bandwidth HF links are utilised.

For most protocol/address type pairs, it is expected that the checksum will be calculated according to the ISO 3309 (HDLC) specification [ISO 3309].

6.6 Comments

As can be seen, the DUAL packet format is very minimal. As an example, where IP is used over DUAL on a Class-C subnet, the overall DUAL link header is 5 octets: 1 for protocol/address, 1 for source, 1 for destination and 2 for the checksum. This compares well with the 19 octets required for the AX.25 header when transmitting IP data, a saving of 74% for the link header.

7 Protocol Ids

DUAL can support up to 32 higher layer protocols. The following protocol ids are currently defined. A descriptive name for each address id/format is also given, starting with the prefix 'PR_'. Note that not all 32 values have been 'given' to protocols, nor has the packet formats for those protocols with protocol ids been defined. These packet formats will be defined in future papers.

Value	Description
0: PR_BCAST	The data field contains some information to be broadcast to all stations on the link. Section 10 describes the format of packets with this protocol id.
1: PR_VC	This id is reserved for future use for a protocol that provides AX.25-style virtual circuit connections.
2: PR_NETROM	This id is reserved for future use in carrying NETROM data.
3: PR_ROSE	This id is reserved for future use in carrying ROSE data.
4: PR_IP	The data field contains an IP datagram, and is described in detail in Section 8.
5: PR_CIP	The data field contains a TCP/IP segment which has been compressed, and is described in detain in Section 9.
6: PR_IPX	This id is reserved for future use in carrying Novell IPX data
30: PR_EXP0	This id is reserved for future use in carrying data for experimental protocols.
31: PR_EXP1	This id is reserved for future use in carrying data for experimental protocols.

At the time of writing, only the PR_BCAST, PR_IP and PR_CIP packet formats and protocol definitions have been defined.

8 Transporting IP Over DUAL – PR_IP

One of our original desires in designing a new Link layer was to provide a better Link layer for the transmission of TCP/IP packets over amateur packet radio. Apart from the already described goals of reducing headers, we also wanted to remove the need for using ARP and RARP. To do this, we decided to use full or partial IP addresses as the Link address when transmitting IP packets. Partial IP addresses can be used when the size of the subnet serviced by the amateur packet link is known.

For both the protocol values **PR_IP** and **PR_CIP**, the size and format of the source and destination addresses are as described in the next section. Note that the source and destination addresses specified in these fields denote the source and destination of the packet on the *current* link, and may be different to the original source of the packet or its final destination.

8.1 Addresses For IP

Instead of using link layer addresses when transmitting IP datagrams, partial or full IP addresses themselves can be used. This obviates the need for any ARP/RARP protocol. DUAL defines the following 8 address types to be used when IP is the higher layer protocol.

Value	Description
0: AD_0IP	The address field is zero octets in length, i.e non-existent. This is useful when the link is a point-to-point link, and the nodes at each end know the other's IP address.
1:AD_1IP	The address field is unsigned and 1 octet in length. It contains the least significant octet of the IP address.
2:AD_2IP	The address field is unsigned, big-endian and 2 octets in length. It contains the 2 least significant octets of the IP address.
3:AD_3IP	The address field is unsigned, big-endian and 3 octets in length. It contains the 3 least significant octets of the IP address.
4:AD_4IP	The address field is unsigned, big-endian and 4 octets in length. It contains the 4 least significant octets of the IP address.

The following are defined for when extended IP addresses of 64 octets become defined.

Value	Description
5:AD_6IP	The address field is unsigned, big-endian and 6 octets in length. It contains the 6 least significant octets of the IP address.
6:AD_7IP	The address field is unsigned, big-endian and 7 octets in length. It contains the 7 least significant octets of the IP address.
7:AD_8IP	The address field is unsigned, big-endian and 8 octets in length. It contains the 8 least significant octets of the IP address.

For all multi-octet address types (i.e **AD_2IP** to **AD_8IP**), the address is transmitted in big-endian format, in the same manner as IP addresses in IP fragments.

8.2 Broadcast Addresses

For address types **AD_1IP** to **AD_8IP**, the broadcast address has all bits turned on (i.e all 1's).

8.3 The CRC Field

The CRC field is two octets in length, and is calculated over the entire packet (except the CRC field itself) according to the ISO 3309 (HDLC) specification. The field is transmitted in big-endian format.

8.4 Limitations And Disadvantages Of TCP/IP Over DUAL

It should also be obvious that a community must choose an adequate size for the link addresses used, so as to distinguish between all nodes on the link. For example, if the link only had nodes with the IP addresses 44.A.B.zzz, where A and B were constant, then AD_1IP addresses would be adequate to distinguish all nodes. However, if nodes with both 44.A.B.zzz and 44.A.C.zzz used the same link, then addresses of at least size AD_2IP should be used.

No amateur radio callsigns are transmitted as part of TCP/IP transmission. This makes packets smaller, but contravenes amateur radio regulations in most countries. Section 10 addresses this deficiency.

The largest omission is that of routing. DUAL performs **no** routing whatsoever. DUAL believes that the given link is a broadcast link, and all stations on the link can be reached. In many (if not all) situations, this is not the case. We believe that this problem can be solved via appropriate IP netmasks and routing protocols at the network level (i.e at IP).

8.5 Example IP Packets Over DUAL

Consider a Class-C subnet with IP addresses 44.136.7.xxx. The machine using the IP address 44.136.7.129 wishes to start a TCP connection with the machine using the IP address 44.136.7.130.

As the DUAL Link services a Class-C subnet, Link address of the type **AD_1IP** can be used. These uniquely identify each station on the link, and thus no ARP/RARP protocol need be used.

44.136.7.129 commences the connection by transmitting a SYN packet to 44.136.7.130[4]:

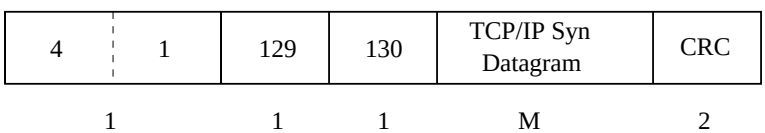

Figure 7: Example Station Identification Packet

Having received this, 44.136.7.130 forms an ACK packet, which is sent back by DUAL:

[4] We will not give full TCP/IP/DUAL packet contents here, but only the details of the DUAL fields.

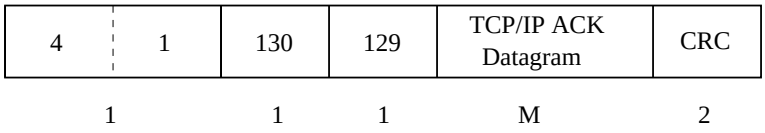

Figure 8: Example Station Identification Packet

The conversation continues on as per the usual TCP/IP method.

9 Transporting Compressed IP Over DUAL – PR_CIP

In the design of DUAL, we believed it was possible to use the TCP/IP header compression scheme used in Compressed SLIP [Jacobson 90]. This would reduce header overhead from 18+40 octets (AX.25, IP, TCP) to 5+5 bytes (DUAL, compressed TCP/IP).

We wished to use the header compression scheme in such a way that the compressed headers did not grow in size, and any alterations to the scheme were small. This has been accomplished. The altered scheme uses exactly the same compressed headers, and the original scheme has not changed dramatically.

The main limitation with the original TCP/IP header compression scheme is that it was designed for use over a slow point-to-point link. Therefore, each end of the link could assume the identity of the machine at the other end. TCP/IP headers are compressed using the method described in section 11.3, and each end keeps state tables of the compression for each TCP connection.

The compressed header sent instead of a normal TCP/IP header is usually 4 or 5 octets in length, and gives the 'differences' between the last TCP/IP header for the TCP connection and the current TCP/IP header. As well, a 'connection number' is sent to the destination, to allow it to find the correct state entry and uncompress the compressed TCP/IP header.

9.1 Protocol Identification

Non-TCP packets with IP headers will be transmitted by DUAL using the **PR_IP** protocol format. Any TCP packets that are given to the header compression scheme for compression will be transmitted using the **PR_CIP** protocol format. This includes uncompressed packets, such as:

- TCP/IP packets which cannot be compressed, as they are the first packets for a TCP connection,
- TCP/IP packets whose state has changed so much that the headers cannot be compressed,
- TCP/IP packets for connections that have 'fallen out' of the LRU state table in the compressor, and
- retransmitted TCP/IP packets needed to be sent uncompressed to resynchronise the uncompressor.

Therefore an implementation of DUAL with **PR_CIP** must also implement **PR_IP**.

9.2 Addresses

All addresses formats for **PR_CIP** packets are exactly the same as for **PR_IP**.

10 Broadcasting With DUAL

All countries have amateur regulations that enforce the periodic broadcast of amateur callsigns. This is satisfied in AX.25 by using callsigns as link addresses. DUAL, however, allows other forms of link addresses. Therefore, a mechanism must be provided to periodically broadcast the node's amateur callsign. This section explains how DUAL satisfies this regulation by using **PR_BCAST** protocol Id.

10.1 Broadcasting Link Addresses – PR_BCAST/AD_CALL

The **PR_BCAST** packet format allows a station to broadcast its callsign and associated link layer addresses to satisfy the amateur regulations. It can also be used for other purposes, as described in the next section.

All broadcast packets include the callsign of the transmitting station as the source address. This is a fixed 10-octet field. The callsign is expressed in 7-bit ASCII, with the most significant bit of each octet off (i.e 0). The callsign is 'big-endian', i.e the first characters of the source callsign come first in the field. If the callsign is shorter than 10 characters, the unused octets have all their bits off (i.e 0). The callsign should be represented by the following characters only[5]:

'A' to 'Z' inclusive, '0' to '9' inclusive, '–' and '/'.

Thus, for example, the callsign VK1XWT would be represented by the 10 octets 86, 75, 49, 88, 87, 84, 0, 0, 0, 0 (decimal).

10.1.1 AD_CALL

When the Protocol Id field is 0 (**PR_BCAST**) and the Address Type field is 0 (**AD_CALL**), the packet contains the sender's amateur callsign and one or more DUAL Link addresses that are to be associated with this callsign. The format of the packet is as follows:

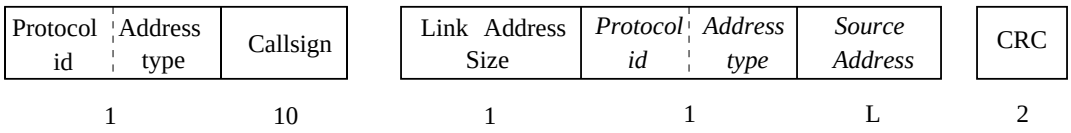

Figure 9: DUAL Station Identification Packet

The packet has no destination address; it is implicit that the packet is to be received by all stations, and hence the destination address is non-existent (or more strictly, it has a length of zero). The Checksum field is 2 octets in length and is calculated as per the checksum in **PR_IP**.

Between the Callsign and the CRC are one or more blocks which give link addresses associated with the callsign. Each block begins with two octets. The first gives the size in octets of the link address in the block. The second octet gives a Protocol-Id/Address-type pair that identifies the format of the given link address.

As an example, on a Class-C subnet where the operator VK1XWT uses the IP address 44.136.7.129, a **PR_BCAST/AD_CALL** would look like:

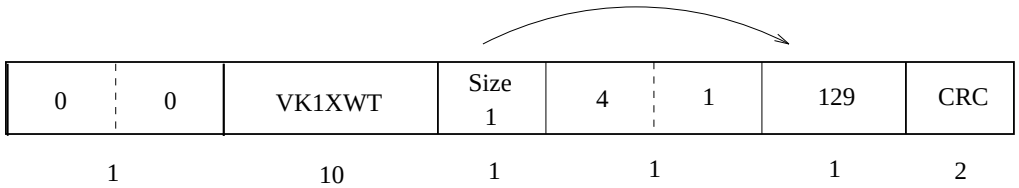

Figure 10: Example Station Identification Packet

The Size field in each block is required, as an implementation of DUAL may not be able to recognise a given link address. Therefore it needs to know the size of the given link address so that it can skip it and parse any further link addresses. The drawback of this field is that it limits link addresses to 256 octets in size.

[5] We can only recommend a character set and a subset of characters to use in English speaking countries. Other amateur services can make their own recommendations.

10.2 Broadcasting Textual Information – PR_BCAST/AD_BEACON

The **PR_BCAST/AD_BEACON** packet format allows a station to broadcast its callsign and plain ASCII text in its Data field. The intention is to allow BBSs to broadcast stuff like: VK1BBS: Mail for VK1XWT, VK10K VK1KCM.

10.2.1 AD_BEACON

When the Protocol Id field is 0 (**PR_BCAST**) and the Address Type field is 1 (**AD_BEACON**), the packet contains the sender's amateur callsign and the BBS broadcast data, for example: VK1BBS: Mail for VK1XWT, VK1OK, VK1KCM. The format of the packet is as follows:

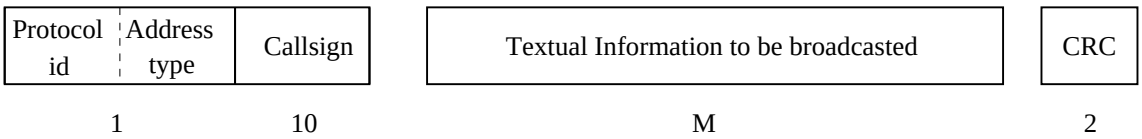

Figure 11: Broadcasting Textual Information

The packet has no destination address; it is implicit that the packet is to be received by all stations, and hence the destination address is non-existent. The Checksum field is 2 octets in length and is calculated as per the checksum in **PR_IP**.

The actual textual information to be broadcasted comes between the Callsign and the CRC. Figure 12 shows how a PR_BCAST/AD_BEACON packet would look like.

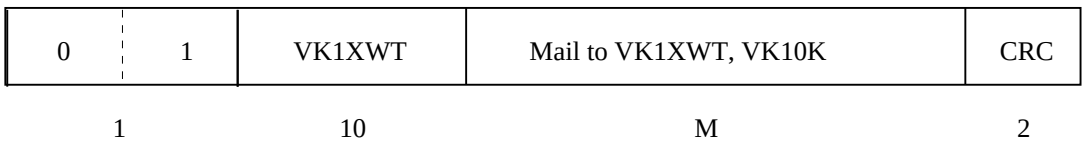

Figure 12: Broadcasting Textual Information

10.3 Callsign Policy

Each amateur service has its own policy on transmission identification with callsigns. Therefore, we cannot mandate a particular callsign policy. DUAL provides a mechanism to broadcast callsigns, to meet each the callsign policy for each amateur service.

11 DUAL Implementation

One of the desires in designing DUAL is to provide a better link layer for the transmission of TCP/IP packets over amateur packet radio. The DUAL protocol format is designed to use a version of the CSLIP, compressed TCP/IP headers, to bring TCP/IP/DUAL packet header overhead further down.

This section will give an overview about how we are going to implement DUAL, and will discuss some of the basic idea and problem behind the compression algorithm. This implementation will use and modifies an existing SLIP/CSLIP implementation. For more information about SLIP and CSLIP see, [Jacobson 90] and [Romkey 88] respectively.

11.1 Implementation Overview

As shown in Figure 13b), the implementation will modify the existing SLIP/CSLIP implementation, Serial Lines IP, which is commonly used for point-to-point serial connections running TCP/IP. The modification is mainly around the packet format which makes DUAL a multi-point protocol, the DUAL packet format is explained earlier in this document.

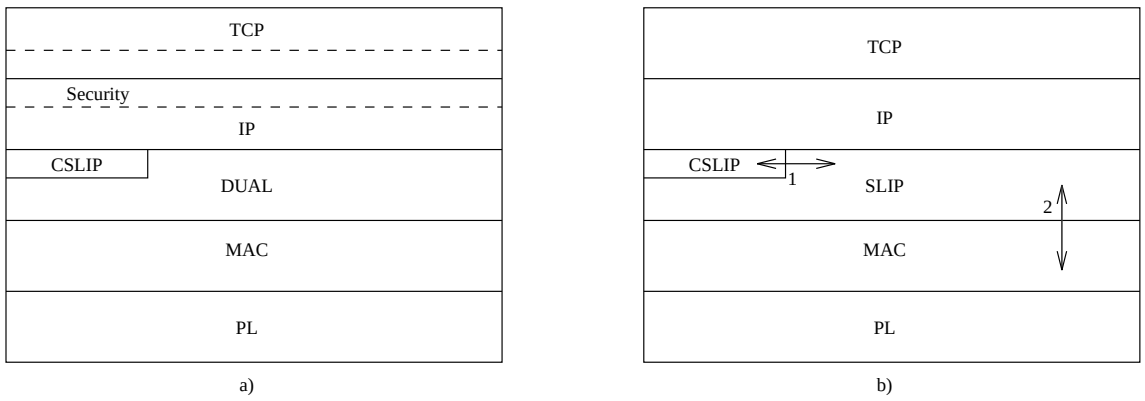

Figure 13: Protocol stack with a) DUAL/CSLIP and b) SLIP/CSLIP

Figure 13b) also shows the interfaces which have to be modified to replace the SLIP with DUAL. Note that most of the modification is going to be within interfaces 1 and 2. Figure 13a) shows how the new protocol stack will look like with DUAL and CSLIP.

The security portion of the protocol stack is shown with dotted line in figure 13a). This part will be implemented after the DUAL is operational. The security portion is a sublayer at the IP layer, and will be implemented using the IP Security Protocol (see [Ioannidis & Blaze 93] for details on the swIPe proposal).

IPSEC provides confidentiality, integrity and authentication of network traffic, and can be used to provide both end-to-end and intermediate-hop security. It is planned to incorporate an implementation of swIPe (see [Ioannidis & Blaze]) with simple policy and key management facilities. It is hoped to develop the key management aspects for large networks and have the granularity for the policy management.

11.2 Implementing DUAL In The 4.x BSD Network Stack

The DUAL protocol, with support for TCP/IP and compressed TCP/IP, has been implemented within the BSD Network Stack in the FreeBSD 1.1 operating system. We chose this system for several reasons:

- The operating system was freely available, stable, sophisticated and flexible. Full source code for the system was available. The operating system runs on i386/i486 machines, which are common.
- The BSD Network Stack is mature and used widely in the Internet community. Nearly all Unix network stacks are derived from the BSD Network Stack. Again, full source code was available.
- The authors had some knowledge of the design and implementation of the above systems. After our implementation, we now have a much better understanding of both.

As DUAL is a Link Layer (or more strictly as LLC layer), it was implemented as an 'interface' in the network stack. We wanted DUAL to produce/receive KISS frames to be able to interface with KISS-mode TNCs, and as the KISS framing scheme is nearly identical to the SLIP framing scheme [Romkey 88], we began with the SLIP interface code.

After some consideration of the existing code in the BSD Network Stack, we realised that there were existing pieces of code that could be shared by SLIP and DUAL, and also PPP. These are:

- The actual SLIP framing code, in `net/if_sl.c`,

- The TCP/IP header compression code, in `net/slcompress.c`, and
- The HDLC checksum code, in `net/if_ppp.c`.

Our first implementation task was to separate these functions in such a way that DUAL, SLIP and PPP could share them. The SLIP framing code was moved into a new file, `net/if_slframe.c`, with the remainder of SLIP left in `net/if_sl.c`. The HDLC checksum code was moved to `net/hdlc_fcs.c`, with PPP left untouched.

The TCP/IP header compression code in `net/slcompress.c` was already shared between SLIP and PPP, and we have not moved any code from this file, but we have substantially altered the code to make the scheme work as per Section 9.

With the shared pieces of code separated, we created `net/if_dual.c` and `net/if_dual.h` from the SLIP interface code, renamed all variables, functions and structures, and began to implement the DUAL protocol.

11.3 The Compression Algorithm

This subsection will describes a method for compressing the headers of TCP/IP datagrams to improve performance over low speed serial links. Note that this compression is specific to TCP/IP datagrams; UDP/IP datagrams are too infrequent to be worth the bother, see [Jacobson 90].

11.3.1 Why Compression Is Desirable

Internet services one might wish to access over a remote serial IP link range from interactive terminal type connections (e.g., telnet, rlogin, xterm) to bulk data transfer (e.g., ftp, smtp). Unfortunately, this extension exposes some complex problems in link-level framing, address assignment, routing, authentication and performance. Header compression is motivated by the need for good interactive response. I.e. the line efficiency of a protocol is the ratio of the data to header+data in a datagram. If efficient bulk data transfer is the only objective, it is always possible to make the datagram large enough to approach an efficiency of 100

Human-factors studies[Sheniderman 87] have found that interactive response is precieved as bad when low-level feedback (character echo) takes longer than 100 to 200 ms. For example if the line is too slow, it may be impossible to fit both the headers and data into a 200 ms window: One typed character results in a 41 byte TCP/IP packet being sent and a 41 byte echo being received. The line speed must be at least 4000 bps to handle these 82 bytes in 200 ms. From this fact it is clear that one design goal of the compression should be to limit the bandwidth demand of typing and ack traffic to at most 300bps[6].

Another design goal is that the compression protocol be based solely on information guaranteed to be known to both ends of a single serial link. Note that a physical link treated as two, independent, simplex links (one each direction) imposes the minimum requirements on topology, routing and pipelining.

11.3.2 The Basic Idea

Figure 14 shows a typical (and minimum length) TCP/IP datagram header. The header size is 40 bytes: 20 bytes of IP and 20 of TCP. Unfortunately, since the TCP and IP protocols were not designed by a committee, all these header fields serve some useful purpose and it's not possible to simply omit some in the name of efficiency.

[6] For more detail discussion about this matter, see [Jacobson 90].

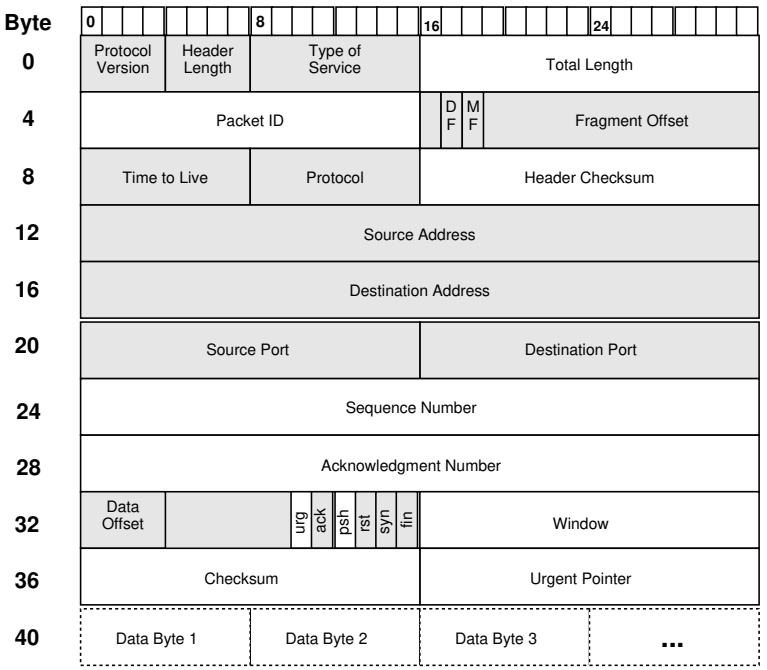

Figure 14: Fields that change during a TCP connection

However, TCP establishes connections and, typically, tens or hundreds of packets are exchanged on each connection. How much of the per-packet information is likely to stay constant over the life of a connection? Half - the shaded fields in figure 4. So, if the sender and receiver keep track of active connections and the receiver keeps a copy of the header from the last packet it saw from each connection, the sender gets a factor-of-two compression by sending only a small (* 8 bit) connection identifier together with the 20 bytes that change and letting the receiver fill in the 20 fixed bytes from the saved hader.

One can scavenge a few more bytes by see how fields changes during a TCP connection life time. If the sender keep a copy of the last packet sent for each connection, it then can figure out what fields change in the current packet, and send a bitmask indicating what changed followed by the changing fields. If the sender only sends fields that differ, this scheme gets the average header size down to around ten bytes. For more detail discussion about this matter see [Jacobson 90].

11.3.3 The Ugly Details

Figure 15 shows a block diagram of the DUAL input/output driver and the compression software. The networking system calls a DUAL output driver with an IP packet to be sent over the serial line. Before passing the packet to the compressor (see figure 15), the output driver builds the DUAL header packet format. Then the compressor checks if the protocol is TCP. Non-TCP packets and uncompressible TCP packets (describe in [Jacobson 90]) are just marked as TYPE_IP and passed to the framer. Compressible TCP packet are looked up in an array of packet headers. If a matching connection is found, the incoming packet is compressed, the (uncompressed) packet header is copied into the array, and a packet of type COMPRESSED_TCP is send to the framer[7].

The framer is responsible for communicating the packet data, type and boundary. The framer encapsulates the incoming packets in KISS format, ready to be transmitted on the radio channel. Since the compression is a differential coding, the framer must not re-order packets. It must also provide good error detection and, if connection numbers are compressed, must provide an error indication to the decompressor.

[7] An UNCOMPRESSED_TCP packet is identical to the original IP packet execpt the IP protocol field is replaced with a *connection number*, an index into the array of saved, per-connection packet headers. This is how sender (re-)synchronizes the receiver.

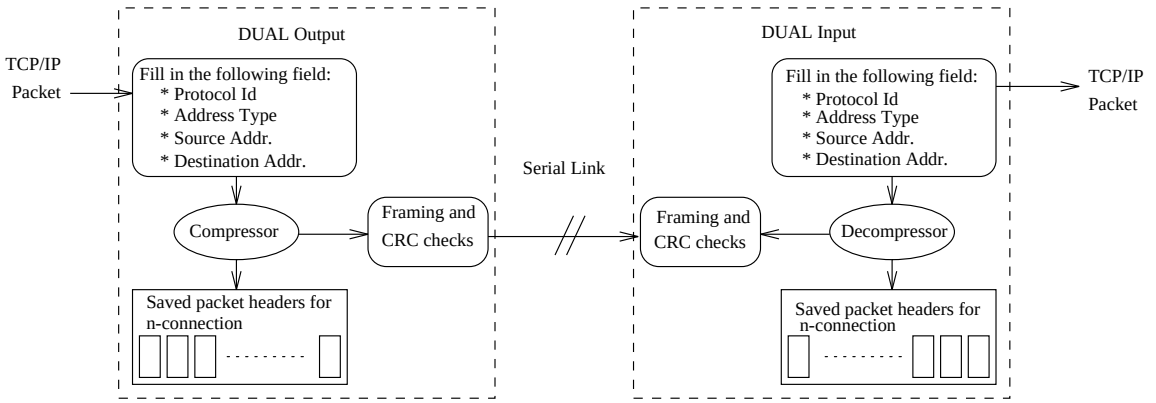

Figure 15: DUAL Input/Output drivers and CSLIP software model

The incoming packet goes through the framer for CRC check, if it failed the framer will discard the packet without notifying the upper layer. The upper layer protocol is responsible for error recovery, i.e. TCP[8]. If the CRC checks does not fail, the framer passes the packet to the decompressor. The decompressor does what the compressor has told it to do, it restores the TCP/IP headers if they have been compressed and then passes the packet to the DUAL input driver to find out which upper layer protocol is waiting for the incoming packet.

Note that the actual compression algorithm is written for point-to-point connection, and because DUAL is a multi-point protocol, in addition to the saved TCP/IP header, we have to save the (partial) IP address too. We need this to identify not only the connection number but also to which machine this packet goes.

11.3.4 Compressed Packet Format

Figure 16 shows the format of a compressed TCP/IP packet. There is a change mask that identifies which of the fields expected to change per-packet changed, a connection number, so the receiver can locate the saved copy of the last packet for this TCP connection, the unmodified TCP checksum so the end-to-end data integrity check will still be valid, then for each bit set in the change mask, the amount the associated field changed. Optional fields, controlled by the mask, are enclosed in dashed lines in the figure.

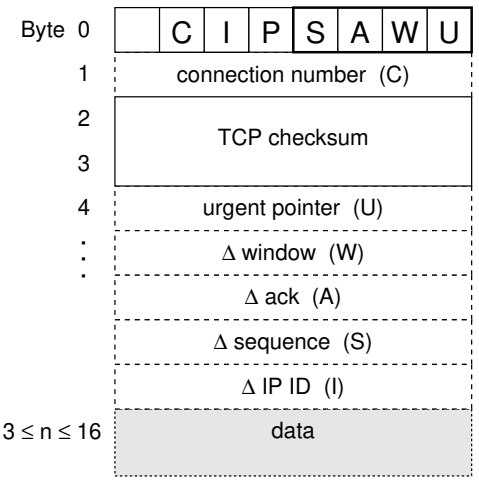

Figure 16: The header of a compressed TCP/IP datagram

[8] The sending TCP will eventually time out when no ACK arrives.

11.4 Modifications To The Compression Algorithm

On a broadcast link, such as DUAL, the connection number in the compressed header is not sufficient to identify the TCP connection. The new scheme saves the **link source address** in each state entry. When uncompressing a packet, the algorithm uses both the connection number and the link source address to find the correct state entry required to uncompress the packet.

At the same time, the state table is stored as a Least Recently Used linked list. This improves searches through the list as the most recently used states will be found quickly. When a new TCP connection is opened, the state entry at the least recent end of the linked list is used to store the state of the connection's compression, and it is of course moved to the front of the list. The original compression scheme used 8-bit connection numbers, but suggested that 16 numbers would be a good implementation maximum over a point-to-point link. In a broadcast environment, a station may have to deal with more than 16 compressed connections, especially if it is a packet router. Therefore, we have increased the implementation maximum from 16 to 256 connection numbers.

One major alteration was a restriction of the compression scheme's behaviour. When two or packets for the same TCP connection are transmitted sequentially by the sender, the connection number is removed from the compressed header. This information is vital in a broadcast environment, so the modified scheme transmits connection numbers in every compressed header, and the receiver rejects any packet without a connection number.

With these modifications, TCP/IP packets broadcast over the link will have their headers compressed, reducing the total header overhead from 58 to 10 octets, an 83% reduction in packet headers.

12 Conclusion

DUAL provides a new format for amateur radio packet transmission that has less header overhead than AX.25 and is more 'user-friendly' to higher-layer protocols such as IP. DUAL provides up to eight packet formats for each upper layer protocol, allowing packet transmission to be tailored to the link environment that will be used.

For the transmission of TCP/IP packets, the header compression scheme used in Compressed SLIP has been altered to work in a broadcast environment. Overall, there is an 83% decrease in header size when using DUAL to transmit TCP/IP packets, compared with TCP/IP transmission over AX.25.

We have implemented a basic DUAL layer in the BSD Network Stack in FreeBSD. The implementation is robust and confirms out protocol design. As yet, we have not completed our implementation to our satisfaction, nor tested DUAL in a real amateur packet radio environment, although we intend to achieve both in a few months. A future paper will describe our results.

13 Further Work

At this stage DUAL has been implemented with the minimum functionality, and is available for testing as described in appendix B.

The following grok is scheduled to be done by the end of this project:

- *Testing:* as stated earlier DUAL is ready for testing. We hope that DUAL will behave properly and we will be able to have some performance result at the end of this stage.
- *Security extension:* more work yet to be done in this area. Hopefully we would be able to implement some security extension at TCP/IP level, possibly with compression, by the end of this project.

A The KISS TNC: A Simple Communications Protocol

The KISS TNC (Terminal Node Controller) provides direct computer to TNC communication using a simple protocol as described briefly here. For more detail see [K3MC & KA9Q 85].

Standard TNC software was written with human users in mind; unfortunately, commands and responses well well suited for human use are ill-adapted for host computer use, and vice versa. This is especially true for multi-user servers such as bulletin boards which must multiplex data from several network connections across a single host/TNC link.

The KISS TNC solves these problems by eliminating as much as possible from the TNC software, giving the attached host complete control over and access to the contents of the HDLC frames transmitted and received over the air. This is central to the KISS philosophy: the host software should have control over all TNC functions at the lowest possible level.

The AX.25 protocol is removed entirely from the TNC, as are all command interpreters and the like. The TNC simply converts between synchronous HDLC, spoken on the full- or -duplex radio channel, and special asynchronous, full duplex frame format spoken on the host/TNC link. Every frame received on the HDLC link is passed intact to the host once it has been translated to the asynchronous format; likewise, asynchronous frames from the host are transmitted on the radio channel once they have been converted to HDLC format.

This means that the bulk of AX.25 or another protocol i.e DUAL must now be implemented on the host system. This is acceptable, however, considering the greatly increased flexibility and reduced overall complexity that comes from allowing the protocol to reside on the same machine with the applications to which it is closely coupled.

A.1 Asynchronous Frame Format

The "asynchronous packet protocol" spoken between the host and TNC is very simple, since its only function is to delimit frames. Each frame is both preceded and followed by a special FEND (Frame End) character, analogous to an HDLC flag. No CRC or checksum is provided. In addition, no RS-232C handshaking signals are employed. The special characters are:

Abbreviation	Description	Hex value
FEND	Frame End	C0
FESC	Frame Escape	DB
TFEND	Transposed Frame End	DC
TFESC	Transposed Frame Escape	DD

The reason for both preceding and ending frames with FENDs is to improve performance when there is noise on the asynch line. The FEND at the beginning of a frame serves to "flush out" any accumulated garbage into a separate frame (which will be discarded by the upper layer protocol) instead of sticking it on the front of an otherwise good frame. As with back-to-back flags in HDLC, two FEND characters in a row should not be interpreted as delimiting an empty frame.

Frames are sent in 8-bit binary; the asynchronous link is set to 8 data bits, 1 stop bit, and no parity. If a FEND ever appears in the data, it is translated into the two byte sequence FESC TFEND (Frame Escape, Transposed Frame End). Likewise, if the FESC character ever appears in the user data, it is replaced with the two character sequence FESC TFESC (Frame Escape, Transposed Frame Escape).

This procedure may seem somewhat complicated, but it is easy to implement and recovers quickly from errors. In particular, the FEND character is never sent over the channel except as an actual end-of-frame indication. This ensures that any intact frame (properly delimited by FEND characters) will always be received properly regardless of the starting state of the receiver or corruption of the preceding frame.

This asynchronous framing protocol is identical to **SLIP** (Serial Line IP), a method for sending ARPA IP datagrams across asynchronous links, except that the first byte of each asynchronous frame between host and TNC is a "type" indicator, describe below.

A.2 Control Of The KISS TNC

Each asynchronous data frame sent to the TNC is converted back into "pure" form and queued for transmission as a separate HDLC frame. Although removing the human interface and the AX.25 protocol from the TNC makes most existing TNC commands unnecessary (i.e. they become host functions), the TNC is still responsible for keying the transmitter's PTT line and deferring to other activity on the radio channel. It is therefore necessary to allow the host to control a few TNC parameters.

To distinguish between command and data frames on the host/TNC link, the first byte of each asynchronous frame between host and TNC is a "type" indicator. This type indicator byte is broken into two 4-bit nibbles so that the low-order nibble indicates the command number and the high-order nibble indicates the port number for that particular command. These type indicators are explained in [K3MC & KA9Q 85]. The type primarily of interest here is type "0" or data frame, which indicates that the rest of the frame is data to be sent on the HDLC channel.

A.3 Using DUAL With KISS Modems

A new Link Layer interface such as DUAL is useless unless it can be easily integrated into an existing system. Fortunately, most existing amateur packet stations use Terminal Node Controllers (TNCs) with KISS functionality [K3MC & KA9Q 85].

All TNCs provides a LLC, MAC and one or more Physical layers, giving the user AX.25 virtual circuit connectivity. TNCs which have KISS support give access directly to the Medium Access Control (MAC) layer, allowing a new LLC layer to be added to the system.

Our implementation of DUAL, described earlier in this paper, encapsulates DUAL packets in KISS format, see figure 17. The computer, which contains the network layers from DUAL up to the applications, can be connected to a TNC in KISS mode, which provides the MAC and Physical layers.

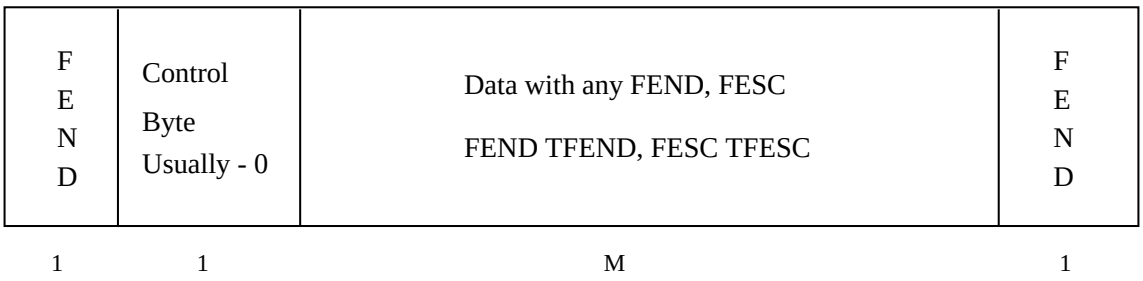

Figure 17: DUAL packets encapsulated in KISS format

B Installation Guide

The following subsections describe how to install the DUAL Link Interface in FreeBSD 1.1. The sample code and a complete Berkeley Unix implementation is available in machine readable form via anonymous ftp from Internet host *minnie.cs.adfa.oz.au* (131.236.23.152), file **DUAL.tar.gz**. This is a gzip Unix tar file. It must be ftped in binary mode. Good luck!

B.1 Introduction

Welcome to the first DUAL Link Interface distribution for FreeBSD 1.1. This code and the installation is **strictly** BETA-TEST, so if you have any problems, try and work it out yourself (especially the installation), and contact Warren Toomey wkt@cs.adfa.oz.au if you can't work it out or you have a suggestion to fix/improve things. You will need:

- An untouched copy of the FreeBSD 1.1 kernel source code.
- A KISS-mode TNC connected to your machine via serial cable.
- Lots of luck! Maybe not, but it's best to have some.

You should have received in this tar file:

Readme	Readme file
Makefile	A Makefile to install the new code
dual_apps.tar	The applications needed for DUAL
new_files.tar	The new kernel source files for DUAL
patches	The kernel files that have changed
docs.tar	Some documentation on the DUAL protocol

Note that the patches file is the most difficult part of the operation; if you have made any substantial changes to the kernel code, the patches will not be applied correctly, and you may have to hand-patch your kernel code to install DUAL.

B.2 Installing The Code

Firstly, unpack the distribution tar file that you have received:

tar vzxf DUAL_BETA.tar.gz

After unpacking the tar file, become root as you will need to be root to install into the kernel source directory. Type: **make -n** to see how the Makefile will install the source code. Note that it places the applications in `/usr/local/src/dual_apps`, and expects the kernel source code to reside in sys.

Change the Makefile if you want the DUAL applications source to go elsewhere, of if your kernel source code is not kept in /sys. If it all looks ok, cross your fingers and type: **make**. Hopefully only the patches file will fail, and if so, you will have to hand-patch your kernel code to install DUAL.

B.3 Building The Code

cd into `/usr/local/src/dual_apps` and type: **make**. This will make the applications needed for DUAL. When done, type: **make -n install**, to see how they will be install; basically, all programs go into `/usr/local/bin` and manual pages go into `/usr/local/man`. If you are happy, then type **make install**.

B.4 Rebuilding The Kernel

At this stage, print out a copy of dual(4) (from `/usr/local/man/man4/dual.4`), which describes how to reconfigure and rebuild your kernel, and how to use the DUAL applications to get DUAL up and running! Have fun, and remember, all questions/ideas can be sent to:

Warren Toomey: **wkt@cs.adfa.oz.au.**

References

[Black 93] Uyless Black. *Data Link Prtotocols.* Prentice-Hall, 1993. ISBN 0-13-204918-X.

[Diersiong & Ward 89] J. R. Diersiong and W. Jeffrey Ward. *Packet Radio in the Amateur Satellite Service. IEEE Journal on selected areas in communications*, 7(2):226–234, February 1989.

[Ioannidis & Blaze] John Ioannidis and Matt Blaze. *The Architecture and Implementation of Network-Layer Security Under Unix.* From AT&T Bell Laboratories.

[Ioannidis & Blaze 93] John Ioannidis and Matt Blaze. *The swIPe Ip Security Protocol*, INTERNET DRAFT, December 1993.

[ISO 3309] ISO 3309. *Data communication – High-level data link control procedures – Frame structure.*

[Jacobson 90] V. Jacobson. *Compressing TCP/IP headers for low-speed serial links*, RFC 1144, February 1990.

[K3MC & KA9Q 85] M. Chepponis K3MC and P. Karn KA9Q. *The KISS TNC: A Simple Host-to-TNC Communication Protocol.* In *Proceedings of the 1995 ARRL Amateur Radio Computer Networking Conference.* Amateur Radio Relay League, September 1985.

[Kahn 77] R. Kahn. *The Organization of Computer Resouces into a Packet Redio Netowrk. IEEE Transactions on Communications*, January 1977.

[Karn *et al* 85] P. R. Karn, H. E. Price, and R. J. Diersing. *Packet Radio in the Amateur Service. IEEE Journal of Selected Areas in Communications*, 3(3):431–439, May 1985.

[Romkey 88] J. L. Romkey. *Nonstandard for transmission of IP datagrams over serial lines: SLIP*, RFC 1055, June 1988.

[Sheniderman 87] B. Sheniderman. *Designing the User Interface.* Addison-Wesley, 1987.

[Stallings 91] W. Stallings. *Data and Computer Communications.* Maxwell Macmillan International, 1991. ISBN 0-02-946478-1, 3rd Edition.

[WB4JFI 84] T. L. Fox WB4JFI. *The AX.25 Amateur Packet Radio Link Layer Protocol.* Technical report, American Radio Relay League, October 1984.