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

Warren Toomey VK1XWT
Lawrie Brown VK1KLB
Omid Mohammadi

Department of Computer Science
University College
The University of New South Wales
Australian Defence Force Academy
Canberra, ACT
December 6, 1994

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 these following parts of the link layer: the user interface (i.e the Logical Link Control), the Medium Access Control mechanism and the Physical Layer – should be separated, and a set of modular protocols for each part be designed for amateur radio use. This will allow the total link layer to be tailored for the medium used and the operating environment, while maintaining a single user interface to the link layer.

A new, simplified Logical Link Control, DUAL, is described, which is not only more suitable than AX.25 for the transmission of higher-layer traffic, but also more bandwidth-efficient.

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
Omid Mohammadi,	Internet e-mail:	omidm@cs.adfa.oz.au

[1] This research is supported by ATERB grant N033/011 1992/1993

Contents

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 *et al* 85] 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

Why the separation of LLC, MAC and Physical Layer is a good idea, and why they should be designed to be modular, so we can mix and match as we need.
With a separation between these layers:

- the interface between the LLC and upper layer protocols need not change if the MAC or Physical layers change.
- The LLC/MAC interface need not change if the Physical layer changes.

3 Previous Protocols – AX.25

A look at AX.25 [WB4JFI 84], why it isn't suitable.

- 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 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.

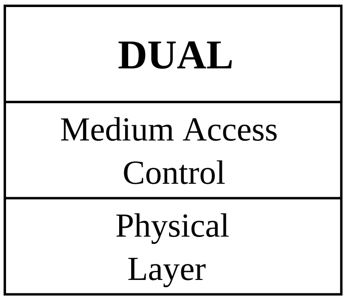

Figure 1: DUAL within the Link Layer

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.

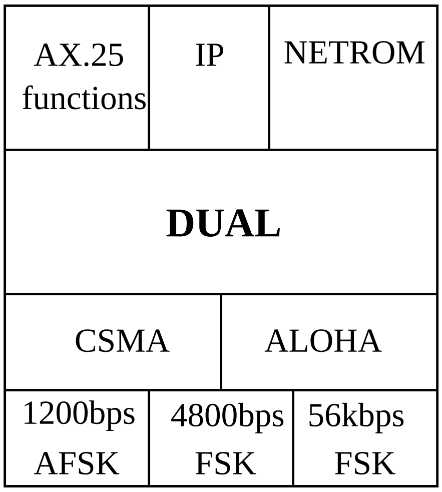

Figure 2: An Example Protocol Stack with DUAL

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[3], 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[4].

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.

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 below:

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

[3] For more informationn about AX.25 see [WB4JFI 84]

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

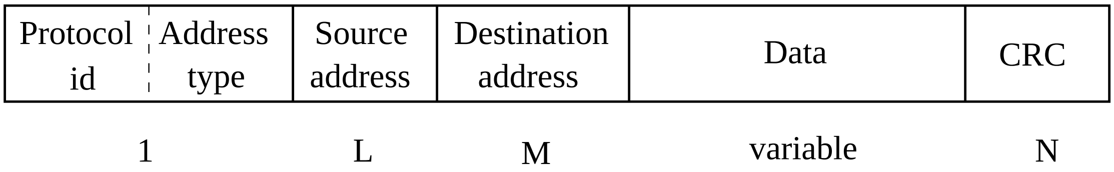

Figure 3: DUAL's Packet Format

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.

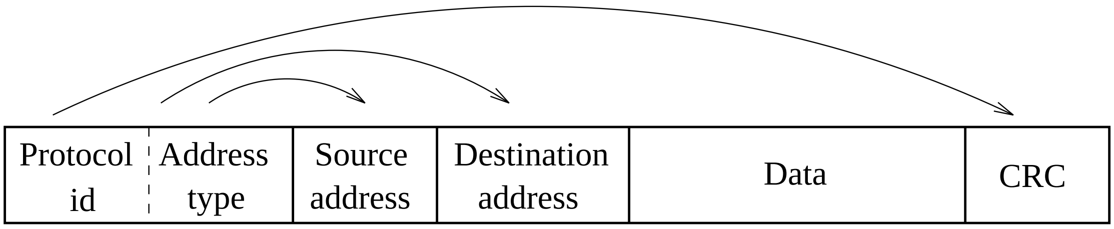

Figure 4: DUAL Field Dependencies

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 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[5]:

4	1	129	130	TCP/IP Syn Datagram	CRC
1		1	1	M	2

Figure 5: Example Station Identification Packet

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

4	1	130	129	TCP/IP ACK Datagram	CRC
1		1	1	M	2

Figure 6: 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.

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

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 Broadcastin 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

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[6]:

'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:

Protocol id	Address type	Callsign	Link Address Size	*Protocol id*	*Address type*	*Source Address*	CRC
1		10	1	1		L	2

Figure 7: DUAL Station Identification Packet

[6] 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.

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:

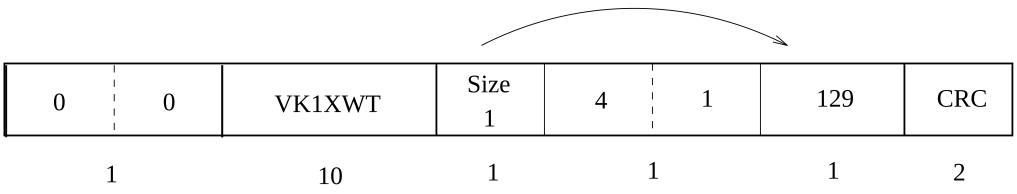

Figure 8: Example Station Identification Packet

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.

10.1.2 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.

10.2 Broadcasting Textual Information

Here the Address field is 1, then the 10-octet callsign. The Data field consists of plain ASCII octets. The intention is to allow BBSs to broadcast stuff like:

`VK1BBS: Mail for VK1XWT, VK1OK, VK1KCM`

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 9b), 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 multipoint protocol, the DUAL packet format is explained earlier in this document.

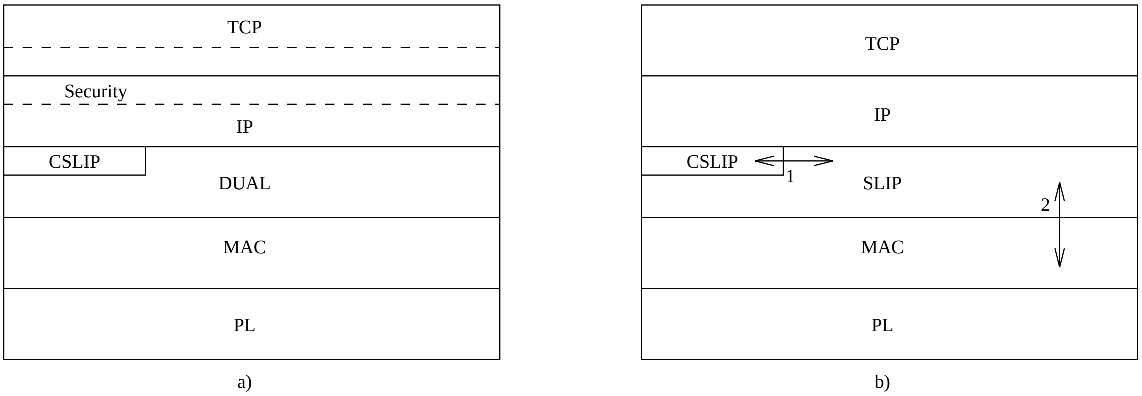

Figure 9: Protocol stack with a) DUAL/CSLIP and b) SLIP/CSLIP

Figure 9b) 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 9a) 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 9a). This part is going to join after the DUAL is operational. The security portion is a sublayer at the IP layer, and will be implemented using the swIPe IP Security Protocol (see [Ioannidis & Blaze 93] formore detail).

SwIPe 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 (see [Ioannidis & Blaze]) of swIPe with simple policay 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 The Problem

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[7].

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.

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

11.3.2 The Basic Idea

Figure 10 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.

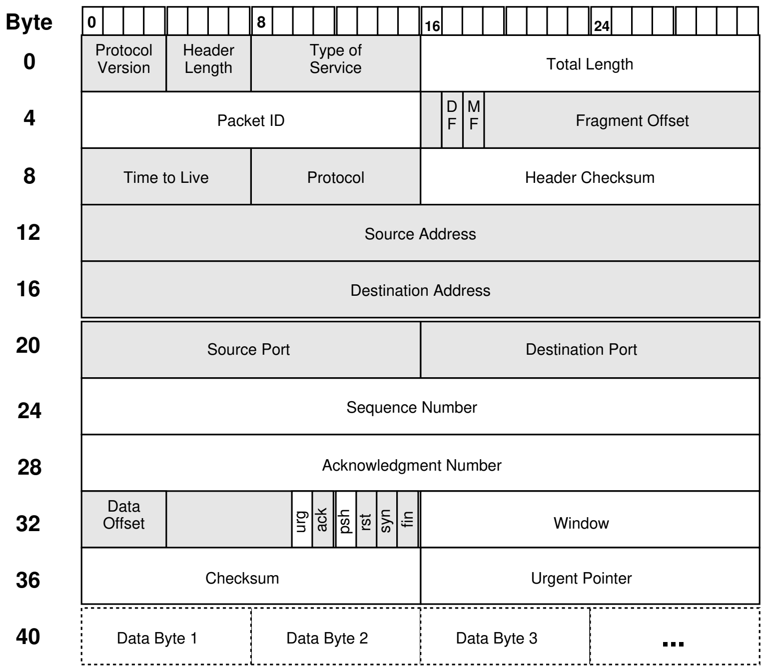

Figure 10: Fields that change during a TCP connection

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 11 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, 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[8].

[8] 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 framer is responsible for communicating the packet data, type and boundary. Since the compression is a differential coding, the framer must not re-order packets (this is rarely a concern over a single serial link). It must also provide good error edetection and, if connection numbers are compressed, must provide an error indication to the decompressor.

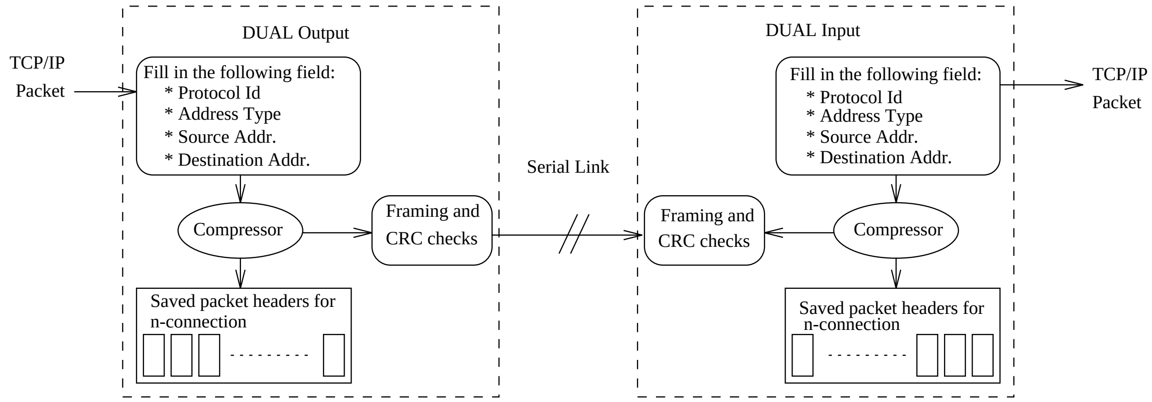

Figure 11: DUAL Input/Output drivers and CSLIP software model

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[9]. 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 multipoint 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 12 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 asociated field changed. Optional fields, controlled by the mask, are enclosed in dashed lines in the figure.

---

the receiver.

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

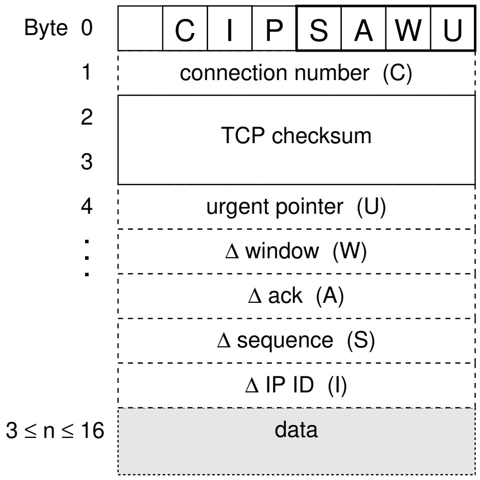

Figure 12: The header of a compres-sed TCP/IP datagram

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.

A 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 in the next section, encapsulates DUAL packets in KISS format. 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.

References

[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 *et al* 85] P. R. Kahn, H. E. Price, and R. J. Diersing. *Packet Radio in the Amateur Service. IEEE Journal of Selected Areas in Communications*, 3(5), 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.

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