A New Communications Paradigm for UNIX

Andrew McRae

Megadata Pty Ltd.
2/37 Waterloo Rd
North Ryde
andrew@mega.com.au

UNIX† and communications have been closely associated since its inception. This paper presents a summary of the development of UNIX communications, and how new paradigms have been created to handle new and changing technology. Recently, digital telecommunications has opened the way for new applications and capability.

The impact of this new technology upon UNIX is discussed, such as what changes are necessary to allow development of new classes of communication applications such as multi-media and integrated fax, and how this technology is fundamentally different to the more traditional networking approach.

Finally a description is given of a development that attempts to provide a working foundation for this new paradigm of digital communications, yet working within the boundaries of vendor supplied UNIX systems, whilst maintaining portability and flexibility.

UNIX and Communications.

UNIX was born and bred in the research department of (what was then) the Phone Company. It is of no surprise then, that communications formed a key part of the early UNIX world, in the form of modem based dial up file transfer and remote command execution (UUCP). It has even been said that UUCP was a device for making more money out of UNIX via the phone bills than could be obtained by license fees. UUCP to this day forms a major portion of UNIX communications infrastructure. Being a store-and-forward system operating over the Plain Old Telephone System (POTS) using readily available modems, it was a good solution for low cost, low bandwidth communications (figure 1). As the widespread distribution of UNIX came about, UUCP was used as the means of connecting these systems into a mesh of interconnected sites. Generally line speeds were limited by the available modem technology, which usually meant 1200 or 2400 baud.

This first phase of UNIX communications spawned early work in the area of networking, electronic mail etc., but left much to be desired. As technology developed, higher bandwidth was available, and new applications were envisaged that were not able to operate across a store-and-forward system. Also the development of high speed Local Area Networks allowed much closer coupling of hosts. In the late 1970's and early 1980's, research work sponsored by DARPA centered around the concept of internetworking, where hosts and networks could be joined together in groups of networks, called a *catenet*, and instead of store-and-forward messaging, datagram technology was used, allowing higher layer protocols to operate seamlessly across a variety of interconnecting media. Work performed at the University of California at Berkeley in the early 1980s allowed a network infrastructure to be integrated into the UNIX kernel, employing the Internet Protocol suite (TCP/IP). Thus the second phase of UNIX communications was heralded, that of internetworking.

† UNIX is a registered trademark of AT&T UNIX System Laboratories.

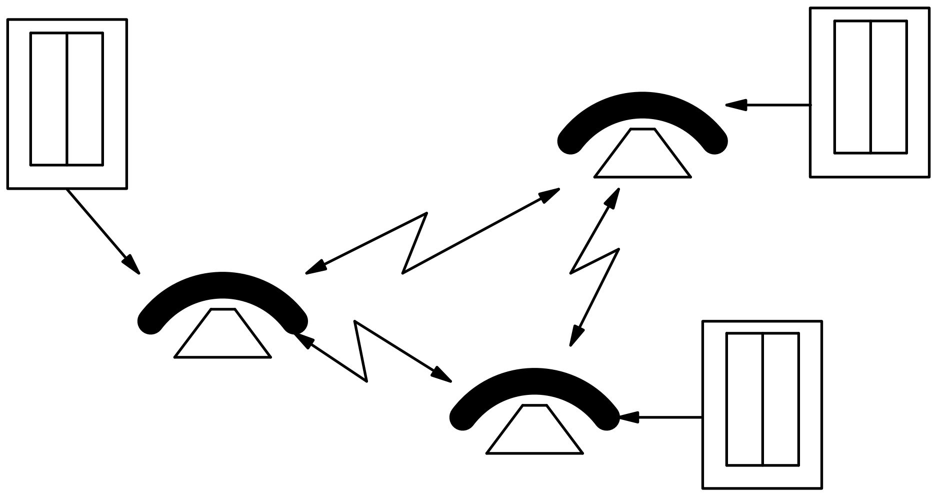
Figure 1: Store and Forward using POTS

Explosive growth resulted, as UNIX systems of all shapes and sizes were interconnected in a vast conglomeration of hosts and networks called the Internet. The common model of this communication infrastructure centred around a group of hosts interconnected via a LAN (within a large site a number of LANs would be used), with connections to other sites via dedicated high speed lines, often via a specialised router. In the short term the configuration was essentially static i.e the interconnecting lines were dedicated, as shown in figure 2.

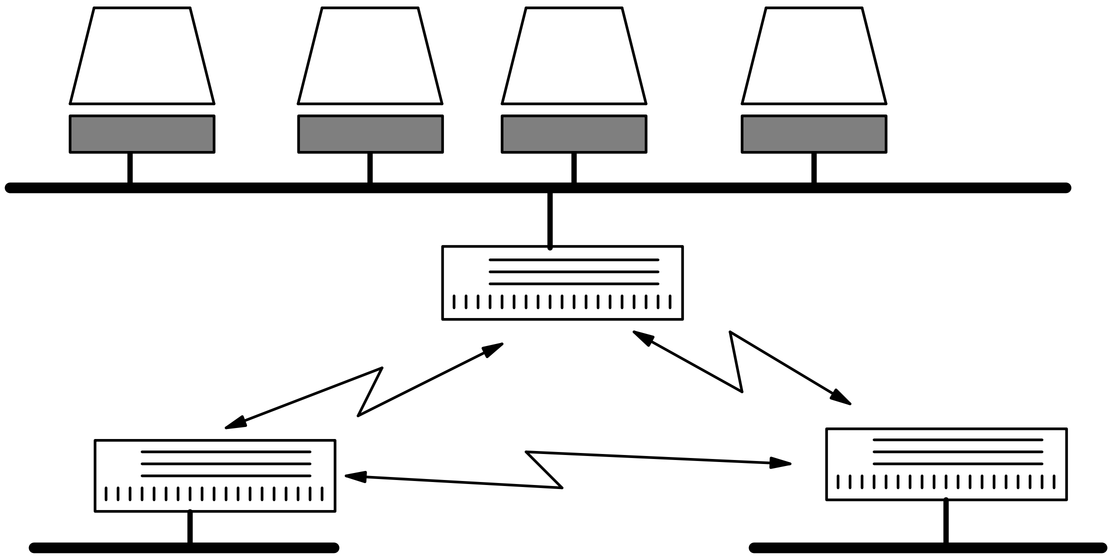
Figure 2: Internetworking

The public (or private) authority providing the communications channel did nothing except provide two endpoints over which data was passed. Other networking technologies have been developed by telecommunication authorities, such as X.25, but these have not been a notable factor except to provide host-to-host dedicated virtual circuits. X.25 has found favour in some commercial arenas precisely for this reason, as a means of connecting users and hosts across a public data network.

Whilst the basic technology of UNIX communications changed from a store-and-forward system to a datagram networking system, the interconnecting lines have not changed greatly, except perhaps in speed and reliability. The *functionality* of the service had not changed from POTS.

The Third Phase.

Latterly, however, a major shift is being experienced in the world of telecommunications, with the advent of ubiquitous digital communications delivered directly to home and office. A number of new technologies are being developed and deployed throughout the telecommunications field, such as Integrated Services Digital Network (ISDN), Frame Relay (FR), Asynchronous Transfer Mode (ATM) etc. These technologies open up exciting new applications and horizons for networking, both by providing orders of magnitude greater bandwidth than before, and also presenting a different service profile to the user.

Just what are these new technologies, and how do they differ from traditional services? In the past, the role of the telecommunications company has been to provide both analogue switched ciruits (POTS) and leased line services. For some time telecommunication companies have been installing and using digital lines for interconnection of exchanges etc., but generally when a line was provided to the customer for use with a computer, a modem was used to convert the analogue line to RS-232 signals. The analogue line back at the exchange was then converted to a (often compressed) digital signal for transfer to its destination; at the destination exchange the digital signal would be restored to an analogue signal, for tranmission to the customer's site (where the modem would convert the signal to a digital RS-232 signal!).

The major change that has occurred in the last few years has been the movement of this Telco/customer digital boundary so that instead of the customer being provided with an analogue signal, the customer is supplied with a direct digital connection, along with all the signalling improvements, thus providing an end-to-end digital service. Being integrated with the infrastructure of the telephone network means that the customer now has access to the full gamut of telecommunication services, such as Fax, voice, data etc. via an integrated digital service.

This digital connection comes in a number of forms, each of which provides for different scales of use and technology. Whilst some of the higher bandwidth options are still being formalised and experimented with, the following are generally available:

ISDN.

The most common is Integrated Services Digital Network (ISDN). ISDN comes in two flavours, Basic Rate Interface (BRI) and Primary Rate Interface (PRI). BRI provides a dual 64 Kbit digital service, along with a 16 Kbit signalling channel. The 64 Kbit channels (known as **B** channels) carry digital data which may be encoded voice, fax information, raw bits etc. The signalling ('**D**') channel is highly structured and standardised, and provides the signalling interface for call setup and tear down, and other related services.

A PRI interface consists of a number of B channels (usually between 23 and 30) and a 64 Kb D channel for signalling purposes. Often a number of B channels are aggregated to provide higher bandwidth, without having to dedicate a complete PRI.

ISDN will be the *lingua franca* of digital communications, with most sites that require low to medium bandwidth employing BRI ISDN or PRI ISDN.

ATM

Asynchronous Transfer Mode (ATM) is a newer technology designed for higher bandwidths, and is based on a small fixed packet length (called a *cell*). ATM is designed to scale to Gigibit networks, and has characteristics that allow allocation of bandwidth on an as-needed basis, and also allows different traffic types to be catered for e.g traffic that can withstand packet loss, but not packet delay, such as real time video.

To allow a more useable interface to ATM, various ATM Adaption Layers (AAL) may be layered on top of the raw cell switching mechanism.

ATM has been touted as the basis for the next generation of ISDN (Broadband ISDN, known as B-ISDN).

FR

Frame Relay (FR) is essentially a stripped down X.25 interface operating at high speeds. It allows network packets to be routed through a common data network similar to X.25, but without the overhead of the upper layers of X.25.

Impact On UNIX.

The current generation of UNIX systems have matured around the internet concept, and a huge variety of applications ranging from simple file transfer (FTP) to graphics (X11) have been developed to operate over TCP/IP, and the catenet model of networking. Furthermore, earlier UNIX systems were generally time-sharing systems, where users would connect via a dumb terminal to a minicomputer, which was connected via LAN to other hosts, and so to the outside world. In the last 4 - 8 years this has been supplanted by the workstation and client/server model, where the unit of computing shrank to the desktop, and each user had their own host which was connected to the LAN. PCs have been traditionally network ignorant, but the distinction between workstation and PC has blurred, and powerful PCs are proliferating on networks and in homes and offices alike.

Part of the shift in UNIX communications has been the growth in single host sites such as homes and remote offices, where dialup modems are used to connect to a central site. The sociological change towards *telecommuting* is increasing, and the supply of good communication technologies will allow this trend to accelerate.

These technologies are *enabling technologies*, and new applications are being introduced rapidly that take advantage of the digital nature of these communication systems. It is clear that UNIX will have to incorporate these enabling technologies somehow, so that it can provide a good foundation for these applications. Already vendors are delivering workstations that are ISDN-capable, allowing applications such as multi-media, digital telephony, integrated fax etc. to be incorporated into the one hardware platform.

Thus we have a new model of communications being incorporated with computing, as shown in figure 3.

How can UNIX best cope with these new technologies, and what transformations need to take place before advantage can be taken from them?

As the digital telecommunications revolution brings the promise of greater bandwidth and fast interconnection to these sites, it highlights the deficiencies in the current UNIX networking model, which traditionally relies on leased lines. Situations are occurring in which users are demanding better and faster services for use with new applications such as multi-media mail, interactive conferencing etc. Coupled with this has been the growth of portable computing; to date networking has been a difficult task for portable computers, but telecommunication carriers have deployed mobile telephony to such an extent that this technology should be made available to support mobile networking. A new paradigm of networking must be developed that will take advantage of these technologies, and integrate them in a rational manner with the current UNIX networking infrastructure.

Rationale for Development.

There is little chance that the issues faced can or will be addressed in the short term evolution of UNIX networking. It will take much research and effort before seamless integration of these applications will occur in commonly available UNIX implementations. Key to these developments will be the rapid and extensive deployment of standardised ISDN. Whilst in some countries this has occurred (and is occurring), there is often a huge installed base of existing infrastructure that would need to be overhauled or supplanted before every person had his or her own ISDN connection to their own home.

Australia is somewhat advanced in the provision of ISDN, having been actively supplying both PRI and BRI ISDN for about 4 years. In most metropolitan areas ISDN is available to the home, at not unreasonable prices (though of course not the same price as POTS).

Having tracked the deployment of ISDN over the last few years, I became interested in how this technology could be put to use within the typical environment of a software developer. Rather than just seeing how ISDN could replace older style leased line technology at a much cheaper rate, I also become

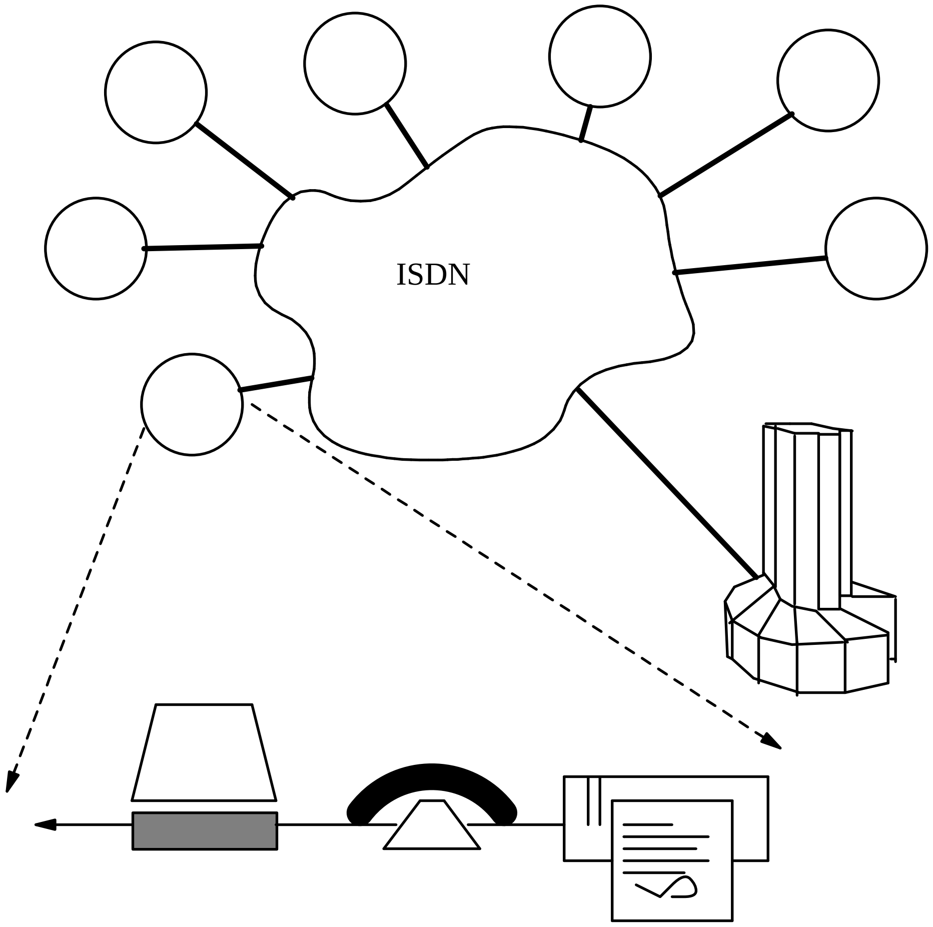

Figure 3: New Model of Integrated Communications

interested in how the more attractive attributes of ISDN could be put to better use. Some of these attributes include:

- Fast connect times, in the order of half a second or less.
- Relatively high bandwidth (compared to modem/POTS technology).
- Reliable digital communications.
- Aggregation of many B channels into a single PRI, each B channel providing separate communication connections.
- A billing model which is time related (but has some call connection costs).

I could see advantages in employing some of these features to provide a low cost but effective solution for the general problem of interconnecting many small systems into an internet, without having to use the existing model of slow dial up lines, leased lines or expensive routers.

It has long been a frustration of many UNIX users that many of the really convenient networking tools such as electronic mail are useful when other parties have access to the tools; so another motivation for experimenting in this field was to find a way of tying in the various disparate communication media (such as e-mail, fax, telephone messages, voice mail) into a more integrated and widespread system.

With mobile computing rapidly becoming a reality, I also saw the need to cater for dynamic control of the network topology, but within the existing Internet framework.

Finally, what I *really* wanted was a good way of telecommuting from home without having to pay for a cisco router and a dedicated line, and also in the future provide for a networked mobile laptop system to take to meetings and impress people.

New Goals.

To cater for the use of these new technologies, especially ISDN, a set of goals were developed:

- Incorporation of other communication facilities, such as FAX or voice mail.
- On-demand allocation of bandwidth, so that the available communications resources could be managed according to priority.
- Support and encourage the use of ‘orphaned systems’, such as workstations and PCs in homes and remote offices, and provide these systems with a much better networking framework than could be provided with UUCP alone, or even regular PPP or SL/IP connection.
- Integrate the older style networking facilities such as dialup SL/IP and PPP over modems.
- Provide intelligent dynamic connection via fast dial-up and tear down.
- Provide facilities that are usually only available in dedicated routers, such as packet filtering.
- Contain facilities to support mobile networking.
- Be portable enough so that it could run on a wide range of platforms, such as home PCs running UNIX, or higher end workstations acting as a communications hub.
- Be aware of and employ to maximum benefit the pricing model of ISDN.

The goals reflect a trend away from dedicated networking equipment for connection to the wider network, and for the use of inbuilt ISDN capability in the actual workstation equipment. A number of vendors are delivering such capability, and it is expected that it will become standard equipment on most workstations, similar to Ethernet.

With these goals in mind, a development was started to explore some of these areas of new technology. The underlying philisopy of this development was not to replace the catenet model of internetworking, but to underpin it with technology that allowed a more flexible use of the digital capability.

The rest of this paper describes this development, and some of the early results.

TUNIP.

TUNIP (derived from TUNnelled IP) was developed initially to provide a more flexible connection strategy for Internetworking over dial up modem lines. The basic goal of *TUNIP* was to provide a means whereby all the disparate elements of communications could be brought together and managed in a portable and flexible way. The plan to do this involved using dial-up ISDN, and a mixture of store-and-forward messaging with on-demand network connection.

The primary goal of *TUNIP* is to entwine the different communication facilities available and to manage them as a whole. In the final version, *TUNIP* will have the following attributes.

- Management of ISDN semi-permanant and dial-up lines, and POTS modem lines.
- Encapsulation of IP using SL/IP, PPP or others as required.
- Dynamic connection to hosts when selected services are requested (such as fax or voice mail).
- Installation of routing table changes to reflect changes in the connection status of hosts.
- Packet filtering to selectively disallow or allow IP packets to particular hosts or ports.
- Reallocation of communication facilities when lines reach predetermined timeouts or in the event of the line not being used.
- On-demand allocation of bandwidth by employing multiple communication lines. This implies both end points must co-operate in providing aggregated bandwidth.
- Backup of ISDN lines via dial up modem lines.

A goal of *TUNIP* is to preserve as much as possible the well known networking tools so that the user is not aware of the underlying network dynamics.

Mobile Networking.

Mobile networking and computing is rapidly coming of age. Much research is being done on ways of supporting mobile networking, especially across the existing infrastructure of the Internet. Part of the problem of supporting this new development is that traditional routing methods rely on a ‘steady-state’ network, where topology changes are relatively few, and are handled by recalculating routes as required. This model doesn’t cope well with nodes *migrating* from one site to another.

If the mobile node is connected via *TUNIP*, when the node dynamically reconnects to another service point, packets routed to the old address will be redirected by *TUNIP* to the new end point until the network routing can catch up. Another method is to allow a *TUNIP* node to act as a *proxy* for the mobile node, so that to the rest of the network the mobile node is only connected at one point, and the packets for that node are redirected to the service point closest to the node by encapsulation and retransmission via the network itself. This effectively hides the mobile nature of that host behind *TUNIP*.

Implementation.

Traditional communication systems under UNIX usually involved kernel level drivers and modules. Performance was often cited as primary reason, but this also made development slow and cumbersome. The trend towards large kernels has been steady, leading to configuration problems, extra memory requirements etc. Microkernels are a response to these problems. Profiling studies I had done previously showed that the user/kernel boundary was not the performance bottleneck previously thought, which opened up the possibility of using one or more co-operating user processes on top of existing kernels.

To achieve the stated goals, *TUNIP* operates as a user level process and interfaces to the kernel via a simple IP tunnel driver, as shown in figure 4. Interface to the hardware communication devices operate through the normal device driver interfaces.

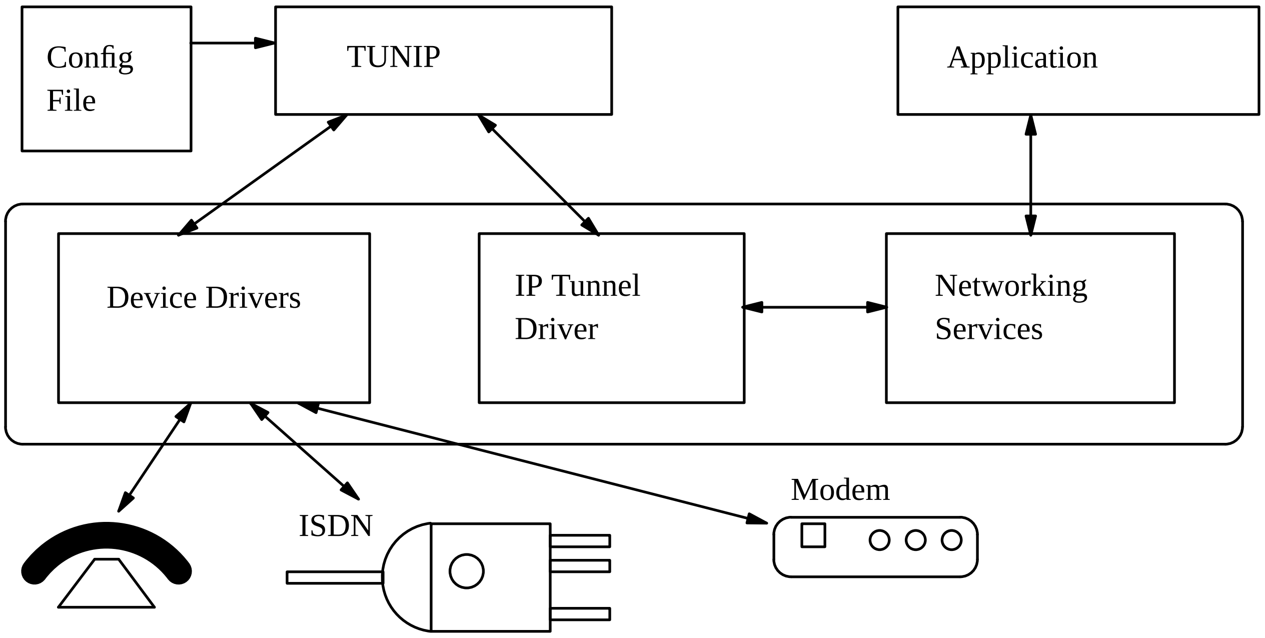

Figure 4: Implementation.

This allows a much greater degree of flexibility and portability than would be otherwise possible, and allowed easy experimenting with different designs, a more robust system (core dumps are much easier to handle than kernel panics), and more flexibility. To a certain extent, loadable device drivers provide similar advantages, but not all systems offer this feature, and there is a certain inelegance to adding even more code to a large kernel.

With most of the code in user space, much more configurability is possible. Each line can be configured to operate as a SL/IP or PPP interface depending on the host to connect to, options such as header compression can be dynamically enabled, there is a simple interface to dialler programs; most importantly, *policy* may be more easily implemented or modified dynamically.

The packet filtering is also flexible, allowing packets to be filtered by port, host or network. Other programs may communicate to *TUNIP* and modify these tables dynamically, or extract statistics.

The question is asked whether this processing belongs in hosts at all, or whether it should be part of dedicated routing hardware. The problem with relying on specialised routers is threefold; cost, mobility and flexibility. With the advent of home systems and mobile systems, having to use a separate piece of hardware to connect these systems is out of the question.

There are a number of other reasons why running *TUNIP* as a separate user process on a host is better than embedding it in a router or other network equipment:

- It is hard to implement a flexible *policy* in a router without timeconsuming or difficult reconfiguration. The policy covers such diverse areas such as security, time accounting, and costing policies to take advantage of different pricing tariffs available at different times of the day.
- Operating as a user process produces a much more portable program, as well allowing enhancements to be developed and tested without kernel reconfiguration. It also eliminates the problem of kernel panics due to coding errors.
- Interaction may occur with other user processes, such as instigating routing changes or user applications upon certain system events (such as starting a mail spooler upon a system connecting).
- Support for sophisticated packet processing such as packet forwarding, encapsulation and retransmission allows development of a framework for mobile networking.

The only specific kernel level software is an IP tunnel driver. This driver appears to the rest of the kernel network software as a network interface driver, configured as a point-to-point interface with a source and destination IP address. Network packets passed to this driver are available to the *TUNIP* user level process via a read/write device; in a similar manner IP packets written to the device will be sent to the kernel network software as if they have been received from the remote host. This simplicity of the tunnel driver reduces the porting effort needed to port *TUNIP* onto a different machine.

The real interface to the rest of the world is via standard device drivers, such as *tty* ports, and other basic level drivers.

Initially the development was done on a Sparcstation under SunOS 4.1.3; I had the advantage that I had done some work on a STREAMS based SL/IP module, which I could benchmark against. The tunnel driver came from a long pedigree, but was publically released by Julian Onions of Nottingham University, and is a simple network interface driver stub accessed by the normal driver read/write interface. Care was taken to make the driver as minimal as possible so that it would be as portable, and all of the policy would be implemented as part of the user process(es).

Porting of *TUNIP* to an Alpha running OSF/1 is under way, and ports to embedded systems have been achieved easily.

TUNIP is initially set up by reading a configuration file, which contains the following data:

- The network interfaces that should be created, and the addresses of the interfaces.
- The pool of line devices it has available, and their characteristics (type, name etc.). Each line may have a *caller* attached to it, similar to UUCP style connections.
- Information about the various remote hosts, such as allowable calling times, phone numbers and security information.
- Protocol filtering information, which allows a packet filter mechanism to be provided.
- Protocol options, such as line encapsulation (SL/IP, PPP), header compression options, protocol prioritisation etc.
- User processes to run upon connection or other triggering conditions.

The configuration file allows all information related to the maintenance of the networking configuration to be installed in one place, which can be reloaded at any time.

Security mechanisms related to networking are often incorporated into routers, such as port and host filtering. Such mechanisms are installed as part of *TUNIP*, which allows tracking and tracing of packets entering the system. Since *TUNIP* may operate on single isolated hosts (without an intervening router), it was important to provide some level of this security.

A sample port filtering configuration is shown below:

```
#
# Port filter
#
protocol allow { all }

udp allow { all }
udp disallow incoming { privileged }
udp allow incoming { echo, discard, time, name }
tcp allow { all }
tcp disallow incoming { privileged, 6000, 6001, 6002, 7000,
7001, 7002,
                        5300, 5330 }
tcp allow incoming
        { smtp, telnet, nntp
#
#       Some extras here to allow remote logins
#
#       1001, 1002, 1003, 1004, 1005
        }
tcp priority { telnet, ftp }
udp priority { time }
```

TUNIP also has the capability of implementing a connection policy reflecting a specified cost model. Dial up ISDN is different from POTS in that it is charged on a timed basis (local telephone calls attract a connection charge, but no time charges). Depending on the granularity of the unit cost, it is often more cost effective to leave a line connected for a minimum time.

As at the time of writing this paper, the ISDN portion of *TUNIP* was in development; ISDN capability was available on the hardware employed, but there has been much delay in approval (and also a lack of software drivers).

Results.

The first phase of *TUNIP* development was completed about April 1993, and has been in service managing a number of SL/IP links over 9600 baud modems. Some of these links are permanent links, and others are dial-up on demand.

The use of a user level process for most processing has certainly been a success, and it is relatively simple to add new devices, diallers, and line encapsulation handlers. The performance of the user process is almost no different from a kernel level system; one serial line on a Sparcstation 1 running at 9600 baud provided statistics as below:

```
Line UP for 153:46:45 - since Sat Jun 19 01:38:41 1993
Statistics last cleared at  Fri Jun  4 14:40:41 1993
                       INCOMING                OUTGOING
Packets via kernel:     3964941                 3116392
Bytes via kernel:    1494654493               137219111
Packets via line:       3964972                 3116392
Bytes via line:      1375303340                47945807
Packets per second:           2                       1
```

```
Bytes per second:           762                        26
Pkts filtered out:           31                         0
Compressed pkts:        3958445                   3116262
Compressed bytes:    1367941407                  41442422
Bytes saved:          126307292 ( 2%)            95768625 ( 7%)
High priority pkts:                                  6818
Line connects:               41
Line drops:                  40
Line errors:               2807
```

As with most Internet links, it is saturated in one direction (incoming, of course). The CPU usage averaged 0.36%, showing that user level processing does not significantly add to the CPU overhead. Scaling the speed of the line or the number indicates that a relatively slow workstation would cope easily with a faster ISDN line, even with packet filtering enabled.

As previously mentioned, the ISDN portion of *TUNIP* is still undergoing some development. Most of the effort expended has been aimed at providing an effective framework for the ISDN to be developed, with the use of modem/POTS technology as a test environment to prove the concept. To this end, the basic ideas of *TUNIP*, such as providing a user level communications system, using a tunnel driver, and allowing several different line types to be accomodated etc., have been proven, and it is expected that as workstation vendors provide approved ISDN solutions, *TUNIP* will be ported to provide a common solution across the complete range of systems.

Conclusions.

TUNIP certainly has achieved the initial goals of providing a more manageable system for supporting integrated communications. It is disappointing that more vendors have not been through the Austel approval cycle for their ISDN interfaces. Once the vendors release workstations with approved interfaces, *and* software to operate them, the range of possibilities opens up greatly. *TUNIP* has been invaluable in helping to explore the cost/benefits of various networking technologies, and also shown that there is much evidence for the benefits of developing such software at the user level as opposed to kernel level.

In some ways the concept of digital services being presented to the customer is reversed from what would be optimal. The model that ISDN presents is of a point-to-point connection oriented communications channel, which is a direct correlation to the older POTS analogue model i.e in some ways ISDN is just a better telephone service. Computer networking, on the other hand, is fundamentally different; that of a host communicating (often at the same time) with a number of other hosts across a network. To allow this behaviour on an inherently connection orientated system is problematic. One either has to emulate a datagram orientated network via fast call connect and tear down (and possibly pay a penalty for not optimizing the connect time), or emulate the exchange model by connecting to a central point(s), which then route the packets to their destination. If the model of networking being used is orientated around a large number of nodes centred around a fewer number of larger centralised sites (such as home systems connecting to the office), the central nodes can employ traditional networking concepts (such as dedicated lines), then a good compromise can be reached, allowing a dynamic and efficient use of the available lines, and also employing a static backbone network.

The same problem exists when dealing with mobile internetworking. There is an existing mobile telephony network, but how is this useable with digital networking technology? One approach is to use modems over the mobile telephone network, but this is clumsy. The main problem is that it is difficult to build a connectionless network over a connectioned infrastructure; if the core telephone infrastructure were packet orientated as opposed to connection orientated, then a connection based telephone service could still operate (over the connectionless network), but the underlying structure could be used in much more effective fashion without the constraints of a connection orientated system.

It can be assured, however, that the phone companies are not going to change the fundamental nature of their infrastructure so it *doesn't* look like a telephone network.