

A Network Architecture Providing Host Migration Transparency

Fumio Teraoka, Yasuhiko Yokote, and Mario Tokoro

Sony Computer Science Laboratory Inc.
3-14-13 Higashigotanda, Shinagawa-ku,
Tokyo 141, Japan.

{tera,ykt,mario}@csl.sony.co.jp

Abstract

The continued expansion of computer networks and the miniaturization of computers increase the desire to use one's computer in a consistent computational environment, regardless of location and time. In this situation, *host migration transparency* is very important. This paper introduces a new network architecture which provides host migration transparency in large interconnected networks and proposes a protocol based on the *propagating cache* method. We introduce the concept *virtual network* and divide the conventional network layer into two sublayers to realize host migration transparency. *Virtual Internet Protocol*, or *VIP* for short, is derived from DARPA-IP as an example of the architecture. We estimate the overhead of VIP. Host migration transparency can be realized without significant processing and traffic overhead. This architecture also preserves host migration transparency across virtual circuits in radio networks.

1 Introduction

In recent years, computers have become smaller and more powerful. Although current small computers have limited functionality and poor communication facilities, in the near future, workstations functionally equivalent to current desk top machines will be small enough to carry. Also, computer interconnection will be so widespread that the various networks will constitute a world-wide distributed computing environment. Users will be able to use computers and networks from any location: campus, office, or home. The continued growth of computer networks and the miniaturization of com-

puters will increase the desire to carry one's own computer and experience the same computational environment regardless of their location within the large interconnected network.

In such an environment, when a user moves, he unplugs the computer from the local area network, transports it, and plugs it in to the local area network at the destination. If radio networks, such as the digital cellular phone system, become widely available as data links, it will be possible to communicate with other computers even while moving. One of the most important features of such an environment is that it provides users with the same computational environment no matter where a user plugs in his computer. Even if a computer is small enough to carry, a user will not carry it if it requires reconfiguration or significant recompilation to achieve the same computational environment after connection to a different network. From the user's point of view, *host migration transparency* allows the same computational environment on the computer regardless of where he migrates with his computer.

There are telephone systems for moving entities, such as automobiles, vessels, trains, and planes. These systems all rely on a similar mechanism, i.e. the radio cellular mechanism. For example, in an automobile cellular telephone system, the range of the radio cell is 3–5 km in an urban area or 10–15 km in a suburb. The mobile station registers its location with the home memory station so that other stations can call the mobile station by accessing the home memory station. Channel switching during a call is supported in all systems except that of the vessel. The services provided in these systems correspond to those of the data-link layer in the OSI seven-layer model. In order to use these systems as data links in large interconnected computer networks, a new network architecture, especially the network layer architecture, is necessary.

Neither the OSI network architecture [ISO 84] nor the existing protocols such as those in the DARPA protocol

suite, take host migration into consideration. Although several distributed systems, such as Mach [Accetta 86], V-system [Cheriton 88], and Chorus [Rozier 88], provide network transparency, these systems also do not support host migration.¹ In these distributed systems, users must consider the locations of roaming hosts when communicating with them. Host migration can be defined as changing the network to which a host is connected. Host migration results in a change in the host address. From the system's viewpoint, host migration transparency can be defined as the service that provides migration independent host identifiers. There are several possible methods to trace migrating hosts, e.g. the name server method and the broadcast method; however, these methods have some problems. According to the network architecture based on the OSI seven-layer model, the network layer is responsible for being aware of host locations within the interconnected networks; thus it is this layer which should provide host migration transparency.

This paper introduces a new network architecture providing host migration transparency and gives an overview of a protocol based on this architecture. Section 2 discusses what host migration transparency is and which layer should provide it. In Section 3, we introduce the concept of a *virtual network* to implement host migration transparency and divide the network layer into two sublayers to efficiently realize the virtual network. In Section 4, we propose a virtual network protocol based on the *propagating cache method*. Section 5 gives an overview of VIP derived from DARPA-IP, as an example virtual network protocol. In Section 6, we compare methods for providing host migration transparency and estimate the overhead of VIP. Section 7 describes the transport layer protocol which preserves virtual circuits for moving hosts. Section 8 concludes this paper.

2 Host Migration Transparency

2.1 What is Host Migration Transparency ?

It can be assumed that a large interconnected network consists of many connected subnetworks, or local area networks (LANs) and that a host is connected to one subnetwork at a time. In such an environment, host migration can be defined as the migration of a host from one subnetwork to another. Host migration results in a change of the network address of the host. Because unplugging a host from a subnetwork (or an *area* in the OSI IS-IS routing model) and plugging it in to another point in the same subnetwork does not change the network address of the host, this kind of move is

¹Amoeba supports host migration only within subnetworks reachable by a broadcast packet [Tanenbaum 91].

not thought of as host migration. We classify host migration into two types: *off-line* and *on-line*. In off-line migration, the host is unplugged from the network before moving and can be used as a standalone machine during transport. On-line migration, in contrast, allows a host access to the network communication channels and reliable connection-oriented mode communication even during actual movement. In conventional wired networks, only off-line migration is possible due to physical restrictions. However, if radio networks, such as the digital cellular phone system, become widely available as data links, on-line migration will also be possible. In this paper, hosts which can migrate on-line are called *mobile hosts* and hosts which can only migrate off-line are called *portable hosts*.

For the user, host migration transparency means that his machine offers the same computational environment regardless of where he chooses to connect his machine to the networks. For example, whether a user plugs his computer in to a network in Tokyo or in New York, he should be able to receive his e-mail on his computer. Such services are offered to the user by application entities, which must communicate with other application entities to continue their computation. If a host migrates, application entities on the host also migrate.² If the network can conceal this migration, users can communicate with each other without any modifications. From a program's viewpoint, host migration transparency means that application entities are not conscious of host migration, and can communicate without modifications.

2.2 Who Should Provide Host Migration Transparency ?

There are several methods to keep track of migrating hosts. For users, name servers provide a kind of migration transparency: an application entity can get the new address of a migrating peer entity through a query to the name server, after which it can communicate with that peer. However, this method has several problems. First, it is impossible to provide on-line migration transparency without modifying application programs. If a host with a radio data link roams into another radio network, the address of the host changes. The peer application entity needs to know that the host has moved and must get the new address of the roaming host to preserve communication channels. Second, this method increases network traffic. It requires communication both between name servers and application entities for queries and possibly among name servers themselves to update the local data caches. Third, it is not clear when to update local caches at each host.

²Process/object migration, i.e. the migration of a process/object from one machine to another, is beyond the scope of this paper.

A broadcast mechanism offers users another kind of host migration transparency. For example, a host which wants to send a packet to a migrating host broadcasts a query packet. The migrating host receives the query and returns its current address to the sender. The sender can then send the packet to the migrating host. One of the most serious defects of this method is that it cannot be used in large interconnected networks, where broadcasting causes a prohibitive increase in traffic.

In both methods, application entities must be aware of the host location. Even if the library routines can conceal the host location, the application layer must still trace the migrating hosts. That is, application entities must be aware of changes of target location for every transmission and get the new address of the target if it has moved.

According to the network architecture of the OSI seven-layer model, host-to-host communication is a network layer service, while end-to-end communication is the responsibility of the transport layer. In other words, the transport layer and upper layers do not perceive the notion of hosts and it is up to the network layer to be aware of host location in the interconnected networks. In conformance with that architecture, the concealment of host location should occur in the network layer. If the network layer provides the upper layers with host migration transparency, application entities do not need to be modified and on-line migration is possible.

2.3 What are the Key Services ?

End-to-end services are provided by the transport and lower layers. To offer host migration transparency in end-to-end services, the transport layer must provide the following four services in addition to the normal services.

- (T-1) Migration transparent transport entity addressing: provides the upper layer with the migration independent identifier of the transport entity.
- (T-2) Migration transparent connectionless mode communication: offers the transport level connectionless mode communication regardless of migration.
- (T-3) Migration transparent connection establishment: establishes a transport level connection regardless of migration.
- (T-4) Migration transparent connection-oriented mode communication: preserves the transport level connection during actual movement and offers the transport level connection-oriented mode communication.

Off-line migration transparency, which must be provided to both portable and mobile hosts, is provided

by T-1 through T-3. On-line migration transparency, required by mobile hosts, requires all four services.

To support these services in the transport layer, the network layer must offer the following six services in addition to the normal services. We assume connectionless mode network services.

- (N-1) Migration transparent host addressing: provides the upper layer with a migration independent identifier of the host. The transport layer can specify the target host by the same host identifier regardless of the host's location.
- (N-2) Migration transparent connectionless mode communication: offers the network level connectionless mode communication regardless of migration.
- (N-3) Disconnect request: prepares to leave a sub-network when a host is to be unplugged.
- (N-4) Disconnect indication: called by the lower layer to prepare to leave the current radio network and enter another in on-line migration.
- (N-5) Connect request: prepares to join a network when a host is plugged in.
- (N-6) Connect indication: called by the lower layer to prepare to join a new radio network in on-line migration.

T-1 is supported by N-1. T-2 and T-3 are supported by N-1, N-2, N-3, and N-5. T-4 is supported by N-2, N-4, and N-6.

In the network layer, N-3, N-4, N-5, and N-6 are supported by the lower layer. Therefore, N-1 and N-2, *migration transparent host addressing* and *migration transparent connectionless mode communication*, are peculiar to the network layer and can be thought of as key services. The next section introduces a new network architecture to provide these two services.

3 Virtual Network

3.1 Concept

In the existing network architectures, host addresses have a hierarchical structure, for example:

- (a) area address + ID + SEL
- (b) network number + host number

In OSI [ISO 90], the network layer address has format (a), where *area address* indicates the routing domain, *ID* is a unique number in the area, and *SEL* is a selector for the transport entity which is to receive the packet. In XNS [Xerox 81] and DARPA-IP [SRI 81a], the host address has format (b). In XNS, the host number is unique across the system and the network number is

added to facilitate the routing. In DARPA-IP, the network number is unique across the system and the host number is unique in the network. In general, it can be assumed that a host address has the two level hierarchy as in (b). In networks employing such host addresses, if a host migrates to another network, the host address changes. Since application entities must be conscious of changes in the host address, host migration transparency cannot be offered under present conditions.

We introduce the concept of a *virtual network* to enable host migration transparency in the network layer. Virtual networks are logical networks which exist virtually on the actual, or *physical*, networks. Virtual networks logically construct interconnected networks above the physical networks. Only the virtual networks are visible from the transport layer. Each host is connected to a virtual network just as it is connected to a physical network. A host never migrates in the virtual networks even if it migrates in the physical networks. That is, each host is always connected to the same virtual network, called the *native network* of the host. A host has a *virtual network address (VN-address)* in the virtual network and a network address, which can be called the *physical network address (PN-address)*, in the physical network. Since a host never migrates in the virtual networks, its VN-address never changes. The PN-address of a host indicates the location of the host in the physical networks and is used for packet routing. The transport layer specifies the target host by its VN-address no matter where it migrates in the physical networks. Basically, the PN-address is invisible from the transport layer. The VN-address and PN-address have the same format.

The relationship between the virtual network address and the physical network address of a host is analogous to that between the virtual address and the physical address in the virtual memory system. In the virtual memory system, the virtual address used in a program never changes no matter where the program is located in physical memory. Similarly, a host can be indicated by its virtual network address no matter where the host migrates in the physical networks.

3.2 Protocol Layering

Since the transport layer specifies the target host by its VN-address, the network layer must convert the VN-address of a host to the corresponding PN-address in order to deliver packets. The network layer is responsible for host-to-host communication, including packet relaying, routing control, and so on. Since this address conversion is not in the traditional category of basic network layer functions, we divide the network layer into two sublayers as follows (see Figure 1):

- The *virtual network sublayer (VN-sublayer)* conceals the physical location of hosts in the physi-

cal networks. A host is assigned a virtual network address (VN-address), which this layer converts to the corresponding physical network address (PN-address).

- The *physical network sublayer (PN-sublayer)* is the conventional network layer. According to the OSI model, this sublayer consists of the subnetwork independent sublayer, the subnetwork dependent sublayer, and the subnetwork access sublayer. This sublayer is responsible for packet relaying, routing information exchange, and so on. A host is assigned a physical network address (PN-address).

According to this separation, the VN-sublayer header is added between the (physical) network layer and transport layer headers. If the (physical) network layer header of a incoming packet specifies the VN-sublayer as the next upper layer, the packet is passed to the VN-sublayer. Otherwise, the packet bypasses the VN-sublayer and is passed to the transport layer. Thus, the VN-sublayer can be bypassed if a incoming packet does not have the VN-sublayer header or if the receiving host does not have the VN-sublayer.

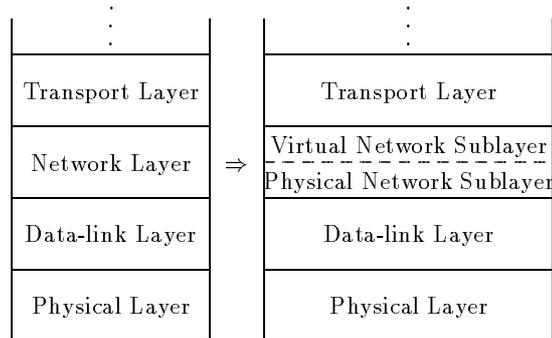


Figure 1: Protocol Layers

4 The General Virtual Network Protocol

This section gives an overview of the general protocols of the virtual and physical network sublayers which do not depend on specific network layer protocols. In [Teraoka 91a], the general protocol is defined using Pascal-like language and a state transition diagram. One of the main services in the VN-sublayer is address conversion as described in Section 3.2. We make the following assumptions about the PN-sublayer: it is the conventional connectionless mode network layer, routing control is executed in the PN-sublayer using the PN-address of a host, and conventional error control occurs in the PN-sublayer.

4.1 The Propagating Cache Method

There are several possible methods for address conversion, such as conversion server method and broadcast method. In the conversion server method, the VN-sublayer of the sender host converts the target VN-address before transmission using a query to the conversion server. In the broadcast method, the VN-sublayer of the sender broadcasts the target VN-address. The target host receives this broadcast packet and replies with its PN-address. These methods, however, have the same defects described in Section 2.2.

To make the overhead of address conversion as small as possible, we introduce the *propagating cache* method. This method falls under the category of lazy evaluation. In this method, each gateway/host (or IS/ES in OSI terms) holds a cache for address conversion. If the sender host does not know the destination PN-address, it assumes that the target host is physically connected to its native network, i.e. the sender assumes that the target's PN-address is equal to its VN-address. The packet transmitted then travels to the native network of the target. If a gateway on the path holds a cache for the target host, the address conversion is done at that gateway. Then, the packet is forwarded to the actual physical location of the target.

Basically, the relationship between the VN-address and the PN-address of a host is maintained by the hosts or gateways in its native network. In the worst case, the packet sent to the migrating host is forwarded by its native network. Many packets, however, are forwarded by gateways before they reach their native network, since cache information propagates to other gateways as communication progresses (see Section 4.2). In addition, the sender learns the PN-address of the migrating host when it receives a reply from that host. Then, it can send packets directly to the migrating host. The propagation of cache information is due to two causes: transmission of control packets of the VN-sublayer and transmission of normal data packets. Control packets are transmitted when a host is connected to or disconnected from a network. Since the virtual and physical network addresses of the sender are included in the headers of both packet types, gateways snoop the header and update local caches when relaying packets.

The cache held in each distinct gateway or host is called an *address mapping table (AMT)*. When a gateway or host receives a packet whose source PN-address is different from its source VN-address, the AMT entry for the source host is created/updated. An AMT entry consists of the following four fields:

- Virtual network address (AMT_{VnAddr}): acts as the key to the entry.
- Physical network address (AMT_{PnAddr}): the requested value.

- Source address timestamp (AMT_{SrcTS}): used to determine whether this entry is obsolete.
- Idle time (AMT_{idle}): used to delete unused entries.

Since the network layer is divided into two sublayers, the network layer header includes both the VN-sublayer header and the PN-sublayer header. The mandatory fields of the VN-sublayer header are as follows:

- Source and destination virtual network address ($VN_{SrcAddr}$ and $VN_{DestAddr}$).
- Packet type (VN_{type}). This field specifies whether the packet is a data packet or a control packet.
- Source address timestamp (VN_{SrcTS}). The source host sets the transmission start time in this field. The value of this field is copied into the AMT_{SrcTS} field of the AMT entry for the source host.
- Destination address timestamp (VN_{DestTS}). This field is filled or modified when the destination VN-address is converted to its PN-address. If the value of this field is older than that of the AMT_{SrcTS} field of the AMT entry for the destination, the PN-address field in the PN-sublayer header is overwritten with the value held in the AMT entry.

In order to determine whether the AMT entry for the destination is obsolete, the virtual network protocol compares two timestamp fields: the VN_{DestTS} field of the packet and the AMT_{SrcTS} field of the local cache entry. This does not imply that the clocks of the hosts in the system must be synchronized because these two timestamps originate from the same host. If the clock of each host in the system increases monotonously, this obsolescence check works correctly.

Transmission, reception, and relay procedures are as follows. If the source host has the AMT entry for the target, it sets the destination PN-address field of the packet the value held in the AMT entry. Otherwise, the source assumes that the target PN-address is equal to its VN-address. The transmitted packet travels by hopping from gateway to gateway. At each gateway, the AMT entry for the source host is created/modified if necessary, the destination PN-address in the packet header is modified, if necessary, according to the AMT entry for the destination, and the gateway relays the packet. The destination host creates/updates the AMT entry for the source if necessary.

4.2 Host Migration

Figure 2 depicts packet flow at the time of host migration. In the figure, an ellipse is a subnetwork, a small circle is a host, and a rectangle is a gateway. The figure shows *Host-X* migrating from *Net-A* to *Net-G*. Assume that the native network of *Host-X* is *Net-A*.

In this migration, the PN-address of *Host-X* changes while its VN-address remains constant. After migration,³ *Host-X* sends a *ConnectionNotification* packet to its native network, *Net-A*. This packet includes its VN-address and its new PN-address. It travels via *Net-G*, *Net-C*, and *Net-B*. Thus, *Gw-CG*, *Gw-BC*, and *Gw-AB* learn the new PN-address of *Host-X*. The packet finally reaches *Net-A* and then is broadcast within *Net-A* to notify hosts in *Net-A* of the new PN-address of *Host-X*. *Gw-AB* returns an acknowledgment (ack) packet to *Host-X*. In the figure, a rectangle marked with a cross represents a gateway which knows the new PN-address of *Host-X*.

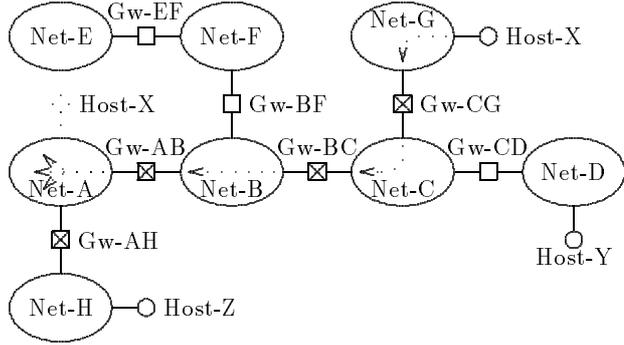


Figure 2: Plug-in to a network

The packet forwarding procedure is depicted in Figure 3. In the figure, *Host-Y* knows the VN-address of *Host-X* but does not know its new PN-address. Therefore, *Host-Y* sets the PN-address field of the packet equal to the VN-address of *Host-X*. The packet heads for *Net-A* since *Net-A* is the native network of *Host-X*. When the packet reaches *Gw-BC*, *Gw-BC* converts the VN-address of *Host-X* to the corresponding PN-address. Then, the packet is forwarded to *Host-X*. Similarly, if *Host-Z* does not know the new PN-address of *Host-X*, the packet reaches *Net-A* and then is forwarded to *Host-X*. In the figure, a thin dotted line shows the path of a packet with an incorrect destination PN-address, and a thick dotted line shows the path of a packet with a correct destination PN-address. If *Host-X* sends a packet to *Host-Y*, *Gw-CD* and *Host-Y* both learn the PN-address of *Host-X*. Thus, as communication between *Host-X* and hosts in other networks progresses, the cache information for *Host-X* propagates more widely. Both the AMT entry and the VN-sublayer header have timestamp fields, so obsolete AMT entries do not convert a destination VN-address to an obsolete PN-address. This prevents routing loops.

Figure 4 illustrates packet flow when a host is unplugged from a network. In the figure, *Host-X* is about to be unplugged from *Net-G*. *Host-X* broadcasts a *Dis-*

³Procedures executed at the time a host is about to be unplugged from a network are described below.

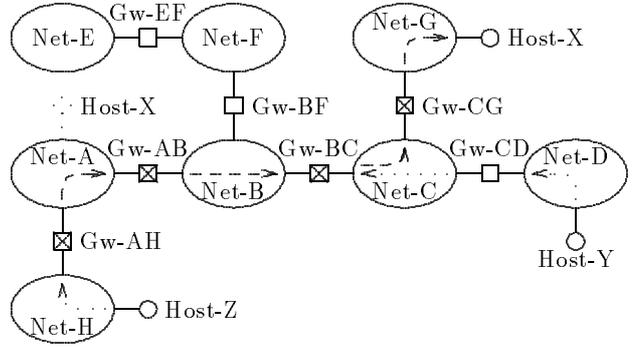


Figure 3: Packet forwarding

connectionNotification packet in *Net-G*. This packet includes the VN-address of *Host-X*. When *Gw-CG* receives this packet, it deletes the AMT entry for *Host-X* and broadcasts the packet in *Net-C* because *Gw-CG* had an AMT entry for *Host-X*. Similarly, *Gw-BC* broadcasts the packet in *Net-B*. *Gw-BF*, however, does nothing when it receives the packet because it does not have an AMT entry for *Host-X*. Thus, most AMT entries for *Host-X* are deleted when *Host-X* is about to be unplugged. This control packet does not travel unrelated networks. The state transition diagram of the VN-sublayer is shown in Appendix A.

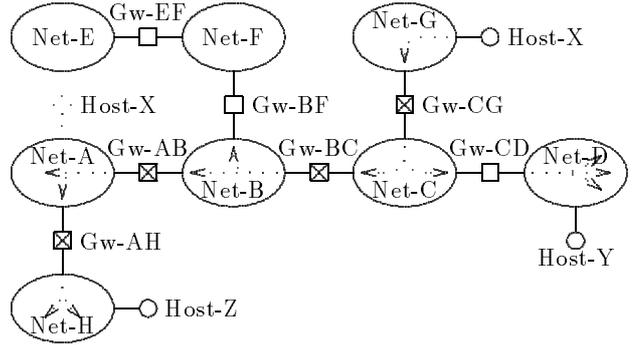


Figure 4: Unplug from a network

4.3 Obsolete Cache Entries

When a host is about to be unplugged from a network, the *DisconnectionNotification* packet is sent and most AMT entries for the host are deleted. Some obsolete cache entries, however, might remain. A gateway or host which finds a packet with an obsolete PN-address returns an *ObsoleteCache* control packet to the source. As the *ObsoleteCache* packet travels, any gateway or host along its path with a corresponding AMT entry deletes that entry. Also, if an AMT entry at a gateway or host is not accessed for a certain time, it is deleted from that cache, with the exception of AMT entries in the native network of the host, which are never deleted.

4.4 Tolerance against Network Partition

In the worst case, a packet sent to a migrating host is forwarded by its native network. The failure of a gateway which partitions the current location of the migrating host from its native network (e.g. *Gw-AB* in Figure 3), might cause a problem. There are two cases: *Gw-AB* may either crash before or after *Host-X* is plugged in to *Net-G*. In the first case, since connection has already completed, information for *Host-X* has already propagated to *Gw-CG* and *Gw-BC* when the partition occurs. In the second case, the *ConnectionNotification* packet transmitted by *Host-X* cannot reach its native network. However, some gateways, e.g. *Gw-CG* and *Gw-BC*, do receive the packet. Therefore, hosts on the same side of the network partition can communicate with the migrating host.⁴ However, the native network cannot learn the new PN-address of the migrating host. This problem can be solved by having the migrating host continue to retransmit the *ConnectionNotification* packet at certain intervals until an ack is received.

5 The Virtual IP – An Example

This section gives an example of the virtual network protocol. *Virtual IP* (*VIP*, for short) [Teraoka 91b] is derived from DARPA-IP. The conventional IP layer is divided into two sublayers: the *VIP sublayer* and the *IP sublayer* (see Figure 5). The functionality of the IP sublayer is the same as that of the conventional IP layer. The VIP sublayer provides host migration transparency.

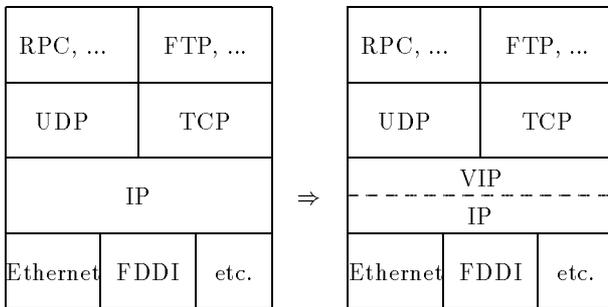


Figure 5: VIP layering

5.1 IP address and VIP address

As described in Section 3.1, a host has two addresses: a VN-address and a PN-address. The *VIP address* is introduced as the VN-address. The conventional IP address acts as the PN-address. The format of both is the same. If a host migrates, the IP address changes while the VIP address is constant. The transport layer specifies the target host using the VIP address so that it

⁴In the Internet, routing control protocols such as EGP must be modified.

can communicate with the target regardless of the target's migration. The IP address, as the PN-address, is invisible from the transport layer in the VIP environment. Each host has a network to which it is normally connected, the native network of the host. The VIP address is originally set to the physical host address in the native network. Thus, originally the IP address and the VIP address have the same value in the native network.

When a host is plugged in to a network other than its native network, it requires a new IP address. The protocol for such IP address allocation is not described in this paper. This allocation can be done with a reverse-ARP like protocol. That is, each local area network has an allocation server to which a migrating host broadcasts an address request. The allocation server then allocates an IP address from the address pool.

5.2 Header Format

The VIP packet header proposed in this paper is implemented as an option of the IP packet header. Therefore, even if a gateway does not implement VIP, it can relay VIP packets. The VIP header consists of the following eight fields (see figure 6):

- Option type ($VIP_{OptType}$) and Option length (VIP_{OptLen}). These fields are defined in IP for options. The $VIP_{OptType}$ field is set to a value specifying VIP. The VIP_{OptLen} holds the length of the VIP header including these two fields (= 20 bytes).
- Protocol (VIP_{proto}). This field indicates the upper layer protocol, i.e. TCP or UDP.
- VIP type (VIP_{type}). This field indicates the type of the VIP packet. There are five types:
 - *VipData*: A normal data packet.
 - *VipDisc*: This packet announces that the source is about to be unplugged from the network.
 - *VipConn*: This packet announces that the source has just been plugged in to the network.
 - *VipConnAck*: This is an ack packet of the *VipConn* packet.
 - *VipErrObs*: This is an error notification packet specifying that an AMT entry is obsolete.
- Source VIP address ($VIP_{SrcAddr}$) and Destination VIP address ($VIP_{DestAddr}$).
- Source address timestamp (VIP_{SrcTS}). This field indicates the transmission start time at the source host. When an AMT entry for the source host is updated, this value is copied to the cache entry's AMT_{SrcTS} field. The AMT is updated if the

value in this field is newer than the value in the AMT_{SrcTS} field of the AMT entry.

- Destination address timestamp (VIP_{DestTS}). When the destination VIP address is converted to its IP address, this field is copied from the value in the AMT_{SrcTS} field of the AMT entry for the destination host. Address conversion occurs only if the value in this field is older than the value in the AMT_{SrcTS} field of an AMT entry for the destination.

In IP, when a packet is sent to a subnetwork, the packet might be split into small fragments if its size exceeds the maximum transmission length of the subnetwork. These fragments are reassembled at the destination host. This processing is called fragmentation and reassembly. When fragmentation occurs, the VIP header must be copied as an option into the IP header for each fragment. Although the source and destination IP addresses and the ID field are used to identify fragments in IP, the source and destination VIP addresses are used instead of IP addresses in VIP.

IP defines eight options, all of which are consistent with VIP, except for Loose/Strict Source Routing. Since the notion of source routing is incompatible with that of migration transparency, these options should not be used. These two options can be used only for fixed hosts whose IP addresses are always equal to their VIP addresses.

Eleven message types are defined in the ICMP [SRI 81b] to report control messages to the source host of an IP datagram. When a gateway forwards a packet, if the outgoing subnetwork and the incoming subnetwork are the same, an ICMP redirect packet is sent to the source host. In contrast, a VIP gateway performing address conversion might send a packet to the same subnetwork from which it was received. An ICMP redirect must not be sent in such case since such forwarding is normal.

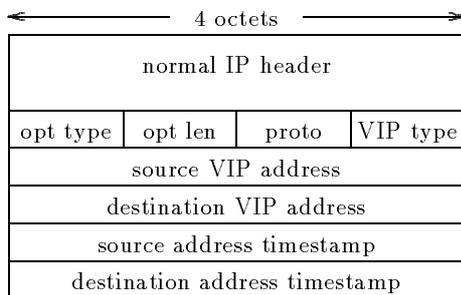


Figure 6: VIP header format

6 Estimation

This section compares the availability, traffic and processing overhead, and tolerance against network partition of the name server, broadcast, and propagating cache methods. This section also quantitatively estimates the overhead of VIP using the propagating cache method. The overheads of VIP compared to IP is negligible.

6.1 Comparison of the Three Methods

Table 1 shows the availability of the three methods. As described in Section 2.2, the name server and broadcast methods cannot offer on-line host migration. The broadcast method cannot be used in large interconnected networks. The propagating cache method can offer off-line and on-line host migration transparency and it can be used in large interconnected networks.

Table 1: Method availability

	off-line migration	on-line migration	wide area
name server	OK	–	OK
broadcast	OK	–	–
propagating cache	OK	OK	OK

Table 2 shows the traffic and processing overhead of the three methods. The traffic overhead of the name server method is due to the registration/modification of the entry and query/reply of the destination address. In addition, if the name server holds the local cache, the cache consistency control among name servers causes traffic overhead. The traffic overhead of the broadcast method is due to the query/reply of the destination address at packet transmission time. In these two methods, it is possible that queries and replies take place at every transmission. The traffic overhead of the propagating cache method is due to the three types of control packets sent at disconnection and connection time. This method does not cause traffic overhead at packet transmission time. This method does cause processing overhead at gateways and the destination host while the other two methods do not, but this overhead is negligible as described in Section 6.2.

The broadcast method is omitted from our estimation of tolerance against network partition since it cannot be used in large interconnected networks. In the name server method, the name server maintains the relationship between the host name and its address. It can be assumed that the host name has a hierarchical structure as in the Domain Name System [SRI 87]. In such a naming system, the host name and address are maintained by the name server in its naming domain. If a network partition occurs between the current location

Table 2: Comparison of overhead

	traffic at plug-in time	traffic at unplug time	traffic at packet xmit	processing at source	processing at gateway	processing at destination
name server	registration to name server	deletion of the entry	query & reply	query & reply	–	–
broadcast	–	–	query & reply	query & reply	–	–
propagating cache	2 control packets	1 control packet	–	AMT entry search	AMT entry search	AMT entry search

of the migrating host and its naming domain, the host cannot register its new address with the name server. Thus, the name server method is not tolerant against network partition. The propagating cache method is tolerant against network partition as described in Section 4.4.

6.2 Overhead Estimation of VIP

In VIP using the propagating cache method, the overhead of packet transmission from source host to destination host in the stable state, T_{o_data} , can be shown as follows:

$$T_{o_data} = T_{src} + T_{gw} \times N_{hop} + T_{dest}$$

where, T_{src} , T_{gw} , and T_{dest} are the processing overhead at the source host, the gateway, and the destination host, respectively, and N_{hop} is the number of hops between the source and the destination. T_{src} , T_{gw} , and T_{dest} can be defined as follows:

$$\begin{aligned} T_{src} &= T_{amt_search} + T_{h_const} \\ T_{gw} &= T_{amt_search} \times 2 + T_{amt_update} \\ T_{dest} &= T_{amt_search} + T_{amt_update} + T_{h_check} \end{aligned}$$

where, T_{amt_search} , T_{h_const} , T_{amt_update} , and T_{h_check} are the time taken for the AMT entry search, the construction of the VIP header, the update of the AMT entry, and the check of the VIP header, respectively. At the source host, AMT entries are searched for the address conversion of the destination address. At the gateway, they are searched for the address conversion of the destination address and for the possible update of the AMT entry for the source host. At the destination host, they are searched for the possible update of the AMT entry for the source host. T_{h_const} and T_{amt_update} consist of several valuable assignments. T_{h_check} consists of several valuable comparisons.

A hash function can optimize the AMT entry search time T_{amt_search} , yielding

$$T_{amt_search} = T_{hash} + T_{compare} \times N_{compare} + T_{key} \times (N_{compare} - 1)$$

where, T_{hash} is the time taken to calculate the lookup key, $T_{compare}$ is the time taken for the key comparison, $N_{compare}$ is the mean number of comparisons, and T_{key}

is the time taken for the generation of the next key if collision occurs. The mean number of comparisons, $N_{compare}$, is as follows:

$$N_{compare} = \frac{1-\alpha/2}{1-\alpha}$$

where, α is the load factor, the fraction of the hash table which is filled. Using linear probing for collision resolution, then α values of 0.25, 0.5, and 0.9 yield $N_{compare}$ values of 1.17, 1.50, and 5.50, respectively. We Assume that the host address $addr$, the hash function $H(addr)$, and the function generating the next key h_{k+1} from h_k are as follows:

$$\begin{aligned} addr &= A1.A2.A3.A4 \text{ (IP's dotted notation)} \\ H(addr) &= A1 + A2 + A3 + A4 \pmod{S} \\ h_{k+1} &= h_k + 1 \pmod{S} \end{aligned}$$

where S is the number of the entries in the AMT.

If the MC68030 CPU with 25MHz clock is used, α is 0.5, and the hop count between the source and the destination is 20, then the data packet transmission overhead T_{o_data} is 0.3–0.4 msec.⁵ The order of the transmission time in the Internet, e.g. between Tokyo and the East Coast of the U.S.A., is 100 msec. Therefore, the transmission overhead in the stable state, T_{o_data} , is negligible.

The memory usage overhead for VIP compared to conventional IP is due to the area for the AMT and the extra area for the VIP header in the packet buffer. The size of an AMT entry is 16 bytes. If a thousand AMT entries are prepared, they occupy only 16 Kbytes. The size of the VIP header is 20 bytes. If a hundred packet buffers are prepared, the memory usage overhead is only 2 Kbytes. Therefore, the extra memory usage of VIP is negligible.

The traffic overhead of VIP is due to control packets. There are three kinds of control packets in the VIP sub-layer besides the error packet: *VipConn*, *VipConnAck*, and *VipDisc*. The *VipConn* and *VipConnAck* packets are transmitted once when a host is plugged in to a network. These packets travel between the current location of the migrating host and its native network. The

⁵For example, the hop count between SonyCSL in Tokyo and MIT is about 20.

VipConn packet is broadcast within the native network. Only two control packets are transmitted when a host is plugged in to a network. Therefore, the traffic overhead at plug-in time is negligible. The *VipDisc* packet is broadcast within the network from which a host is about to be unplugged. The gateways with AMT entries for the host relay that packet. In a broadcast type network such as Ethernet, the cost of broadcast is equal to that of unicast. The *VipDisc* packet is propagated only to those networks which were on the communication route to/from the migrating host. The *VipDisc* packet is not propagated to unrelated networks. Thus, the traffic overhead at disconnect time is negligible.

7 Mobile Host Support

In radio networks, off-line migration transparency can be achieved in the network layer by the method described in Section 4. To achieve on-line migration transparency, the transport layer must preserve the established connections while moving. This section describes the transport layer procedures which preserve the transport connection in on-line migration.

It can be assumed that radio networks consist of many radio cells, each of which can be thought of as equivalent to a conventional (wired) subnetwork as shown in Figure 7. Each cell is assigned a network number. In the figure, a dotted circle is a radio network and the small rectangle at the center of each radio network is a base station or gateway. These gateways are connected by wire. A small circle is a host migrating in the radio networks. We can also imagine the dotted circles in the figure as a cellular phone system consisting of many cells. Such a cellular phone system can be thought of as equivalent to a conventional (wired) subnetwork.

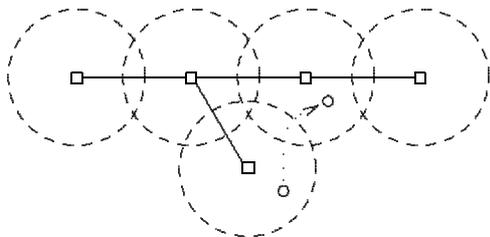


Figure 7: Host migration in radio networks

In addition to the normal control packet types in the transport layer, there must be four extra types of control packets to preserve transport connections: *TconnSuspend*, *TconnSuspendAck*, *TconnResume*, and *TconnResumeAck*. In on-line migration, the data-link layer is responsible for notifying the network layer when the host enters the common area between the current radio cell (or cellular phone system) and the adjacent cell. The procedures to preserve the transport connections are as follows:

1. When a host enters the common area of adjacent cells, the data-link layer notifies the network layer. The network layer then notifies the transport layer. The transport layer changes the state of each connection from *Active* to *Suspend*.
2. The transport layer sends the *TconnSuspend* packet to the peer entity of each transport connection and waits for *TconnSuspendAck* packets. After all acks are received, the transport layer issues the migration request to the network layer.
3. The network layer processes the migration procedures described in Section 4.2.
4. The transport layer sends the *TconnResume* packet to the peer entity of each transport connection and waits for *TconnResumeAck* packets. After all acks are received, the transport layer changes the state of each connection from *Suspend* to *Active*.

The state transition diagram of the transport connection is shown in Appendix B.

8 Conclusion

Telephone systems for moving entities such as automobiles are already available. Also, the construction of local area radio network is underway. In these systems, however, the usage of radio is limited within the subnetwork. That is, the corresponding protocol techniques are part of the data-link layer. The near future will bring the construction of large interconnected radio networks. Because host migration transparency is paramount in such an environment, a new network architecture for radio networks which can handle mobile/portable hosts will be essential.

This paper proposes that host migration transparency should be provided in the network layer according to the OSI seven-layer model and introduces the concept of a *virtual network*. This concept divides the network layer into two sublayers: the *virtual network sublayer* and the *physical network sublayer*. Each host has two identifiers: a migration independent identifier (the *virtual network address*) and a migration dependent identifier (the *physical network address*). To minimize the overhead of address conversion, we introduce the *propagating cache* method. This method falls under the category of lazy evaluation. This paper also proposes VIP derived from DARPA-IP by applying the concept of the virtual network. The overhead of VIP versus IP is negligible. If radio networks become widely available as data links, a host migrating to another network can continue connection-oriented mode communication. VIP is being implemented on the Muse operating system [Yokote 91].

Acknowledgments

We give our thanks to Dr. Hideyuki Tokuda of Carnegie Mellon University, Dr. Jun Murai of Keio University, the members of the WIDE project, and the members of Sony Computer Science Laboratory Inc. Discussions with them helped us refine the concept of virtual network and its adaptation to the IP.

References

- [Accetta 86] Mike Accetta, Robert Baron, David Golub, Richard Rashid, Avadis Tevanian, and Michael Young. *Mach: A New Kernel Foundation For UNIX Development*. Technical Report, Department of Computer Science, Carnegie-Mellon University, August 1986.
- [Cheriton 88] David R. Cheriton. THE V DISTRIBUTED SYSTEM. *Communications of the ACM*, Vol.31, No.3, pp.314–333, March 1988.
- [ISO 84] ISO. *Information processing systems – Open Systems Interconnection – Basic Reference Model*. 1984. ISO7498.
- [ISO 90] ISO. *Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol for use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473)*. February 1990. ISO DP 10589.
- [Rozier 88] M. Rozier, V. Abrossimov, F. Armand, I. Boule, M. Gien, M. Guillemont, F. Herrmann, C. Kaiser, S. Langlois, P. Léonard, and W. Neuhauser. Chorus Distributed Operating Systems. *Computing Systems*, Vol.1, No.4, Fall 1988.
- [SRI 81a] J. B. Postel. *Internet Protocol*. September 1981. RFC 791.
- [SRI 81b] J. B. Postel. *Internet Control Message Protocol*. September 1981. RFC 792
- [SRI 87] P. V. Mockapetris. *Domain names – concepts and facilities*. November 1987. RFC 1034.
- [Tanenbaum 91] Andrew S. Tanenbaum. In his lecture in Sony Computer Science Laboratory Inc. January 1991.
- [Teraoka 91a] Fumio Teraoka. *A General Protocol Specification of the Virtual Network Sublayer*. Technical Memo SCSL-TM-91-001, Sony Computer Science Laboratory Inc., January 1991.
- [Teraoka 91b] Fumio Teraoka. *The Virtual IP Protocol Specification*. Technical Memo SCSL-TM-91-007, Sony Computer Science Laboratory Inc., February 1991.

[Xerox 81] Xerox. *Internet Transport Protocols*. XEROX CORPORATION, December 1981. X SIS 028112.

[Yokote 91] Yasuhiko Yokote, Fumio Teraoka, Atsushi Mitsuzawa, Nobuhisa Fujinami, and Mario Tokoro. The Muse Object Architecture: A New Operating System Structuring Concept. *Operating Systems Review*, Vol.25, No.2, April 1991. also available as SCSL-TR-91-002 of Sony Computer Science Laboratory Inc.

Appendix A

The state transition diagram of the virtual network sublayer is depicted in Figure 8. In the figure, S- n is a state. $\frac{E-n}{O-m}$ means the state is changed by the event E- n and the output O- m is issued. The states, events, and outputs are as follows:

- States:
 - S-1: *Disconnected*: disconnected from the network.
 - S-2: *WaitVnConnReq*: waiting for the VN-sublayer connect request.
 - S-3: *WaitVnConnDoneInd*: waiting for the VN-sublayer connect done indication.
 - S-4: *WaitVnPeerConnAck*: waiting for the VN-sublayer connect notification ack from peer.
 - S-5: *Connected*: connected to the network.
 - S-6: *WaitVnDiscReq*: waiting for the VN-sublayer disconnect request.
 - S-7: *WaitVnDiscDoneInd*: waiting for the VN-sublayer disconnect done indication.
- Events:
 - E-1: *VnConnReq*: the VN-sublayer connect request is called by the transport layer.
 - E-2: *VnConnInd*: the VN-sublayer connect indication is called by the PN-sublayer.
 - E-3: *VnConnDoneInd*: the VN-sublayer connect done indication is called by the PN-sublayer.
 - E-4: *VnDiscReq*: the VN-sublayer disconnect request is called by the transport layer.
 - E-5: *VnDiscInd*: the VN-sublayer disconnect indication is called by the PN-sublayer.
 - E-6: *VnDiscDoneInd*: the VN-sublayer disconnect done indication is called by the PN-sublayer.
 - E-7: *VnPeerConnAck*: the VN-sublayer connect notification ack is received from peer.

- Outputs:
 - O-1: *TConnInd*: calls the transport layer connect indication.
 - O-2: *TConnDoneInd*: calls the transport layer connect done indication.
 - O-3: *TDiscInd*: calls the transport layer disconnect indication.
 - O-4: *TDiscDoneInd*: calls the transport layer disconnect done indication.
 - O-5: *VnPeerConn*: transmits the VN-sublayer connect notification to peer.
 - O-6: *VnPeerdisc*: transmits the VN-sublayer disconnect notification to peer.
 - O-7: *PnConnReq*: calls the PN-sublayer connect request.
 - O-8: *PnDiscReq*: calls the PN-sublayer disconnect request.

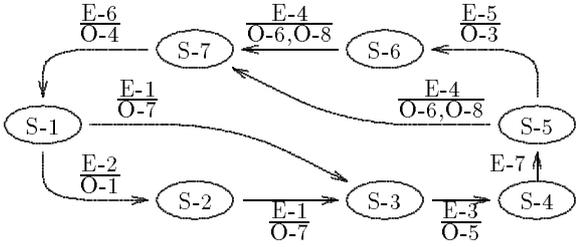


Figure 8: State transition diagram of the virtual network sublayer

Appendix B

The state transition diagram of the transport connection related to host migration is depicted in Figure 9. In the figure, S- n is a state. $\frac{E-n}{O-m}$ means the state is changed by the event E- n and output O- m is issued. The states, events, and outputs are as follows:

- States:
 - S-1: *Connected*: the transport connection is connected and available to transmit/receive packets.
 - S-2: *WaitTconnSuspendAck*: waiting for the *TconnSuspendAck* packet from the peer transport entity.
 - S-3: *WaitTDiscDoneInd*: waiting for the *TDiscDoneInd* to be called by the lower layer.
 - S-4: *Suspended*: the transport connection is suspended and unavailable to transmit/receive packets.

- S-5: *WaitTConnDoneInd*: waiting for the *TConnDoneInd* to be called by the lower layer.
- S-6: *WaitTconnResumeAck*: waiting for the *TconnResumeAck* packet from the peer transport entity.

- Events:
 - E-1: *TDiscInd*: the transport disconnect indication is called by the VN-sublayer.
 - E-2: *TDiscDoneInd*: the transport disconnect done indication is called by the VN-sublayer.
 - E-3: *TConnInd*: the transport connect indication is called by the VN-sublayer.
 - E-4: *TConnDoneInd*: the transport connect done indication is called by the VN-sublayer.
 - E-5: *TconnSuspend*: the *TconnSuspend* packet is received.
 - E-6: *TconnResume*: the *TconnResume* packet is received.
 - E-7: *TconnSuspendAck*: The *TconnSuspendAck* packet is received.
 - E-8: *TconnResumeAck*: the *TconnResumeAck* packet is received.

- Outputs:
 - O-1: *TconnSuspend*: transmits the *TconnSuspend* packet.
 - O-2: *TconnResume*: transmits the *TconnResume* packet.
 - O-3: *TconnSuspendAck*: transmits the *TconnSuspendAck* packet.
 - O-4: *TconnResumeAck*: transmits the *TconnResumeAck* packet.
 - O-5: *VnDiscReq*: calls the VN-sublayer disconnect request.
 - O-6: *VnConnReq*: calls the VN-sublayer connect request.

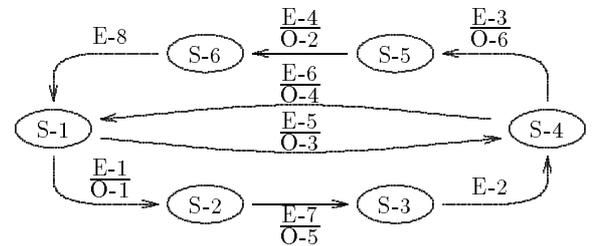


Figure 9: State transition diagram of the transport connection