TIA Subcommittee TR41.5
CONTRIBUTION # TR41.5/97-08-007

PROJECT: Residential Gateway / Home Networks

TITLE: Proposed Standards for Home Network Operating Systems

AUTHOR: Rich Janow

SOURCE: Lucent Technologies Inc.

CONTACT: Rich Janow
Lucent Technologies / Bell Laboratories
600 Mountain Avenue
Murray Hill, N. J. 07974, Room 3d-568
Phone: 908 582-5285
Fax: 908 582-2970
Email: rjanow@bell-labs.com

DATE: August 18-19 1997 (Quebec)

DISTRIBUTION: TIA TR41.5 Subcommittee Working Group

ABSTRACT: This contribution discusses requirements for the Home Network Operating System that will supervise Residential Gateways and appliances. It offers an HNOS architecture, defines key technical constructs and technology items, and discusses principles of operation.

NOTICE: This contribution includes information that has been prepared to assist TIA Standards Subcommittee TR41.5 in their work program. This document is submitted for discussion and working group purposes only, and is not to be construed as binding on Lucent Technologies Inc. or a proposal from Lucent Technologies Inc. Subsequent events may lead to changes in the information contributed in form, content, and/or numerical value. Specifically, Lucent Technologies reserves the right to change its position on, add to, or amend any of the statements and subject matter contained herein.
Proposed Standards for Home Network Operating Systems

A Draft Contribution to TIA TR41.5
10 August 1997
Version 1.1

Dr. Richard Janow
Bell Laboratories
Technology Strategy and Assessment Department
908 582-5285

Lucent Technologies
600 Mountain Avenue
Murray Hill, New Jersey 07974
Forward

Home Networks will appear over the next several years and become mainstream consumer tools for accessing new and traditional multimedia and wireless communications and information applications. These systems must be able to use heterogeneous access systems as information sources, and they must also insulate consumers from discontinuously complex appliance and user interfaces at every level. There will be a growing need to manage groups of home appliances as client-server systems, much in analogy with current computer systems.

Residential Gateways are one component of Home Networks, all of which will beneficially be managed by a Home Network Operating System (HNOS) conceived and executed with enough generality to spur future technical evolution. The technical requirements on such a system are severe, inasmuch as the appliances will be heterogeneous over a wide range in terms of computing power, protocols, support for applications, etc. Few traditional desktop or commercial OSs systems have dealt with such a broad range of platform capabilities.

The HNOS will, among other tasks, beneficially integrate control over home appliances and also allow consumers to exploit competing access networks and service offerings. They will work smoothly with easy-to-use visual and speech processing user interfaces. Given a visionary choice of HNOS standards, disjointed appliances and home networks may evolve to integrated “electronic servants”, in which many component appliances retain for many years their traditional heterogeneous lineage and technological diversity, while the market may offer others that fit naturally into the new home fabric.

The problem is more stressful than the one solved by desktop OS’s, and a new, lightweight, open architecture, distributed approach shall be followed.
Table of Contents

1. DRIVING REQUIREMENTS 4

2. HOME NETWORK FUNCTIONAL SUBSYSTEMS 6
   2.1. User Experience and Application Manager 8
   2.2. Interoperability Broker 8
   2.3. Resource Locator Subsystem 9
   2.4. Home Network Manager 9
   2.5. Access Gateway Manager 10

3. ESSENTIAL CHARACTERISTICS OF AN HNOS 12

4. HNOS SYSTEM ARCHITECTURE 14
   4.1. An NOS Definition 14
   4.2. Technical Approach for an HNOS 14
       4.2.1. Resources Abstracted as Files 15
       4.2.2. Name Spaces 15
       4.2.3. Standard Communications Protocol 15

5. HNOS OPERATION AND COMPONENTS 16
   5.1. The OS Kernel 17
       5.1.1. Native vs. Emulation Kernels 17
       5.1.2. Memory Management 18
       5.1.3. Scheduling 18
       5.1.4. Devices 18
       5.1.5. Name Space Construction 18
       5.1.6. Security 19
       5.1.7. Database Support 19
   5.2. Programming Language 19
   5.3. Universal Virtual Machine 20
   5.4. Transparent Networking 20
       5.4.1. Communications Protocol 20
       5.4.2. Hiding Network Complexity 21
1. Driving Requirements

Over the long-term Residential Gateways (RGs) will become common edge devices for growing, heterogeneous home networks. The RGs, in fact, make home networks in the true sense feasible by isolating the media inside homes from details and interfaces of the access systems extending from the physical up through the network layers. The current situation in which devices are dedicated to a particular access system will change.

- Intelligence will continue migrating to ever more widely distributed appliances and communication devices.
- Particular services will normally be available via more than one access means.
- Access media will simultaneously carry several distinct connections and classes of service.

RGs deal with this freedom and the added complexity. Appliances and communication media inside the home will interconnect strongly to one another and with the RG - either centralized or distributed - potentially forming a well-integrated system of home networks and information appliances that interoperate with one another. A home network operating system (HNOS) is needed to make the interoperation seamless, incorporating RGs as components and hiding the complexity of the system from applications and their users.

The HNOS should therefore be broadly conceived and specified so that it facilitates long-term evolution to integrated home systems. Figures 1.1 and 1.2 suggest the diversifying applications and range of functional capabilities that homes will advantageously need as they develop home networks.

In the interim, RGs are likely to appear with centralized, distributed, and frequently with hybrid architectures. For example, many telephony and high speed data applications can profitably use a centralized RG, while broadcast applications and some very high speed data applications may require a distributed RG architecture for reasons related to industry practice and bandwidth limits. The HNOS shall accommodate these disparate physical means while providing equally convenient use of the diversifying services on each medium. The long-term final state of the physical systems is not easily predictable and may not be the same everywhere. It need not be the same if the HNOS adequately insulates appliances and users from platform-specific complexity. And so the top-level HNOS requirements are these broad ones:

- The HNOS shall be architecturally and algorithmically able to handle the most general and therefore the most stressful case - that of a distributed system including a broad range of clients and servers, in terms of device intelligence, within homes and on the network edge, incorporating both distributed and also centralized RG modules concurrently, given that they adhere to conventions of the HNOS. The clients and servers include all consumer electronic appliances, home automation devices, communication gear, and small computers and peripherals commonly found at home.

- Additionally, the HNOS shall transparently support advanced communications services including network based applications that may download small applications, and electronic commerce applications that may impose security/privacy needs.

As a result, the consumer products and services industry will be able to create offers that use and interwork with appliances, irrespective of the installed physical platforms.

These general constraints dictate the choice of an HNOS design that is explicitly addressed to distributed networked applications using heterogeneous computing nodes and heterogeneous media. One great need is for simple means of interoperability support. Additionally, the HNOS shall support one or several programming languages that can run on heterogeneous platforms, much in the manner of migratory “applets”, using a common API (called the “universal virtual machine”) that the HNOS provides. There must be a user-definable way to name resources and physical endpoints in the network, so that they can be accessed wherever they reside, and a database for storing system self-
knowledge. Communications protocols should be lightweight and transparent. There must be reliable support for security. Additionally, the HNOS shall support residential command and control languages such as CEBUS, X10, and perhaps others that are needed to direct legacy appliances. In the U.S., it needs to be consistent with the emerging CEBUS standard (and in Europe with the EHSA specifications).

The home networks themselves should be seen as a client/server system but with media, interfaces, and some functions different from business systems. Miniature servers can host special processing and storage functions - much as they might in long haul or access networks; they may be distributed about the home, concentrated in several rooms, or centralized in a miniature network service complex.
2. Home Network Functional Subsystems

Home system control architectures fit a common model having five functional subsystems. Their implementations may be distributed or centralized and they may have dedicated assets to control. The HNOS is closely linked to the control architecture - it shall be the “glue” that creates the illusion of “seamlessness” for application programs and users.

The discussion that follows lies partly outside the conventional scope of a network operating system as seen by OS professionals, but it is important as a way to define the basic HNOS structures needed. As remarked earlier, the HNOS must count on lasting heterogeneity - consumers will not readily accept the notion of single vendor environments or rapid equipment obsolescence.

Figure 2.1 shows this architecture, with control generally flowing down the hierarchy from the top of the chart. Each subsystem can request services directly or indirectly from those on the same level or below it. The four systems below the dashed line directly control and manage home system resources down to the physical layers. They include: 

- Broadcast, Multicast Reception
- Networked Computing / Internet Information / Database Manipulation
  - Video, Image, Multimedia, ...
  - Query, Search, Selection, Transactions
- Simulation / Visualization

- Telemetry, Sensing, & Surveillance
- Command & Control
- Actuation, Robotics

- Telephony - Voice, Video
- Messaging - Voice, Data, Video, Integrated Multimedia
- Collaboration - Work and Gaming (Visual, MM)
- Home Intranet, Data Networking, Client Server Computing
- Personal/Wireless Communications

The diagram also highlights the following functional capabilities needed in homes:

- Select & Manage Access Media
- Distribute Video, Voice, & Data Home-Wide
- Simplify, Personalize, Unify User Interactions
- Unify Appliance Interfaces
- Broadcast, Multicast Reception
- Networked Computing / Internet Information / Database Manipulation
- Video, Image, Multimedia, ...
- Query, Search, Selection, Transactions
- Simulation / Visualization
- Telephony - Voice, Video
- Messaging - Voice, Data, Video, Integrated Multimedia
- Collaboration - Work and Gaming (Visual, MM)
- Home Intranet, Data Networking, Client Server Computing
- Personal/Wireless Communications
- Telemetry, Sensing, & Surveillance
- Command & Control
- Actuation, Robotics
- Ease of Use
- Scale Economy
- Control
- Content
- Environment
- Communications

Figure 1.2: Functional Capabilities Needed in Homes
the basic HNOS (Home Network Operating System) plus middleware additions and an associated database. Applications (above the line) may additionally need their own databases, pointing into those maintained by the HNOS.

The Access Gateway Manager and Home Networks Manager talk directly to and may have modules within the RG. A fully integrated home would use all of the subsystems extensively, but some functions may be sparsely utilized in the near-term. For example, the Resource Locator Subsystem may eventually use telemetry that can follow movements around homes and redirect profile-based services, but near-term forms of it would incorporate at least a database that associates resources and terminals with names and that maintains appliance clusters.

RGs make the system scaleable to large numbers of signal sources: they isolate home networks from the details of access systems' physical layer through network layer details. The RGs create a home network that manages its own bandwidth and establishes internal or external connections on demand.

A shared database of distribution facilities, names, and appliance characteristics gives the system the self-knowledge needed to self-configure many functions. It has to be easy for users to change as needed through HNOS-supported system tools. "Plug and play" setup is best, but the embedded base of older devices will persist for many years.

Figure 2.1: Residential Networks Functional Subsystems
2.1. User Experience and Application Manager

The HNOS shall be able to directly interact with consumers through top level interfaces. As Figure 2.1 suggests, it supervises and controls applications running on the home system and presents a seamless, easy to use, consistent, and eventually user-adaptive look and feel. Several lightweight user interfaces shall be provided to let users talk to the system through video, telephony, PC, appliance control terminals. Standard UIs shall include DMTF (touch tone), TV and game controllers with video display, and a small GUI for computers, screen phones, and the like. Others (to be defined) may include server-hosted speech recognition. The GUI may be run as a application under WINTEL and other large PC OS’s.

Applications, network-based or otherwise, shall be able to substitute their own UIs or use the HNOS defaults when they suffice. Figure 2.2 is an end-to-end view of the software architecture, showing the boundary between the HNOS middleware and application domains.

Here is a partial function list:

- Requests and starts home applications and indirectly calls for resources to support them, often from subservient subsystems. Resolves contention (over resources and intent) when necessary using user profile information.
- Selects and controls applications using visual, speech/audio, and traditional button-oriented user interfaces.
- Provides access to consistent HNOS-supported APIs for running content applications in the home (home-based or remotely downlinked executables).
- Supervises security at top-level and prevents intrusion or invasion of privacy. Coordinates activities of physical security subsystem (application), data security (access manager), surveillance sensors, etc. Limits information accessible by persons, vendors, computers, etc. inside, outside, or in attached networks. Keeps personal profiles closely held unless release is authorized.
- Uses simple through sophisticated “agent” technologies as they prove themselves, to adapt to users’ preferences. Provides convenience applications that collectively create an “electronic servant” illusion. Integrates activities of adaptive features in subsystems.

2.2. Interoperability Broker

The Interoperability Broker Subsystem (IB) glues together the disparate components in the home, from physical to application layer. It’s principal task is to:

- Service applications by defining compatible interworking combinations of appliances, servers, and communication media on demand from the Application Manager. These are implemented by the Home Network Manager and Access Gateway Manager.

The HNOS (see below) simplifies the brokering through an underlying file metaphor for devices along with a file directory model for naming. The system builds on this a system resources database that includes capabilities, location and status, interface and special characteristics, classes of service, predefined groups and clusters. The IB System supports multiple, sometimes contending applications concurrently.

Here are some representative Interoperability Broker functions:
Home Network Operating System

- Unifies control structures for appliances and legacy systems (as feasible). Matches control interfaces, brokers protocols, translates command and control languages, and arranges connections and data stream conversion. The appliance database supplies compatibility information.

- Defines fallback procedures for applications with mismatched terminals or servers - i.e., least common denominator modes.

- Unifies data interchange across in-home media and appliances - routing across RG internal gateways, appliance UI and resource matching at application level. Defines processing pipelines to convert data formats (e.g., A/D conversion, use of decoders, protocol matching). Links applications to suitable platforms.

- Sets up communication and control linkages to older, legacy home systems (e.g., pre-existing home security systems with communication capability) where feasible.

- Supervises telemetry/sensing and real-time appliance control.

2.3. Resource Locator Subsystem

The Resource Locator Subsystem (RLS) detects and names resources in the home, and inserts their descriptors into the home resources database used by the IB and applications systems:

- Detects/administers system configurations and keeps them consistent. Supports plug-and-play installation, trouble shooting, updating, and maintenance using remote network intervention and visual techniques. Self-configures operating environment on system startup. Maintains portions of the HNOS database, provides tools for updating the network facilities and appliances database.

In advanced systems, the RLS supports Residential Personal Mobility - an extension of Nomadcity / Mobility notions from businesses and public facilities to homes. This involves the RLS functions above with time dependence: seamlessly detecting and transferring Cellular/PCS connectivity from hand-held to stationary terminals, and also having communication, entertainment, or security services appear magically at stationary or mobile appliances that happen to be convenient to each person. It should not be necessary to hand carry special communications devices around.

- The RPM system uses telemetry to detect users’ locations and identities as they move around the home.

- It redirects applications (communication, content, comfort control) to nearby appliances over media with appropriate capabilities. For example, interactive applications may have differing needs for processing, input devices, displays that have to be met at locations that themselves are differently equipped.

- It may detect returning PCS devices carried into or brought near the home and cooperate with the Access Manager to arrange signaling that forwards communications through wired access or a fixed wireless Access Gateway that injects them onto a home network to make them available at stationary terminals.

- It may share information with and be part of advanced security systems.

Personal detection telemetry has not appeared in consumer mass markets - it must be low cost and unobtrusive so that consumers can live comfortably and informally with it. The initial RLS systems rely on terminal addresses.

2.4. Home Network Manager

The Home Network Manager (HNM) is a miniature network operating system, operating under supervision of the Interoperability Broker. The HNM sets up home-wide any-to-any connections (limited by media) between input
devices, output devices, servers, adjunct processors, etc. They may be connection-oriented or connectionless. The HNM may:

- bridge, route, or switch command and control flows between subsystems and media.
- similarly bridge, route, or switch content flows between subsystems and media.
- provide connections on demand between nodes in the home and Access Gateways in any combination
- set up conversion processing pipelines where needed to match connections, using shared filters and processing resources.
- deliver files (high-speed data, executables, messages) to processors and storage devices.
- support permanent channels (typically shared) that are used by legacy or very low cost home systems.
- signal modems and adapters to switch channels or modes, using internal and/or standard protocols.
- monitor its own performance, resolves contention.

The HNM may be distributed over several home network gateway devices, with part or all of it possibly residing as an organic part of a centralized RG platform, or on a vendor-supplied interface card to an RG, or on a distributed RG element, or on a server that is attached to one of the home networks. The HNM should be viewed as potentially a hierarchy of HNMs, with one of them designated as the root, master node.

Home network manager platforms may carry on-board applications that should be distinguished from HNOS organic functions. For example, a telephony/communications manager may:

- provide voice, video, email messaging support functions, storage, and control. These must work smoothly with TAS (telephone answering system) and messaging applications.
- provides intercom/PBX functions for voice and video telephony.
- incorporate service-provider applications such as access "arbitrage" (similar to least cost routing) by answering call by call requests with the "best" access options, prices, and user features. A local knowledge base defines these.
- provide audio, and possibly video, conference bridging. It may support collaborative workgroup activities that share video/image/data and use voice and possibly video telephony. Applications using this facility include work-at-home, but especially also virtual community activities.

2.5. Access Gateway Manager

The Access Gateway Manager arbitrates and selects among access media and service providers, and coordinates network services that need home CPE. It may exercise command and decision making functions, process data related to service costs and preferences, and operate the network interface devices.

The kinds of access systems foreseen at this time are:

- Twisted pair, carrying POTS, or ISDN BRI, and xDSL modulations.
- One or two way, analog or digital coaxial cable, frequency divided into channels. Currently used for video broadcast, expanding potentially to high speed data, voice, digital video.
- Fixed wireless loop. narrowband media using analog or digital technology related to cellular or PCS.
Home Network Operating System

- Direct broadcast satellite, including a satellite dish and other special video reception equipment. Unlikely to truly have satellite uplink, but terrestrial uplinks (low speed) may appear.

- In the long term, broadband microwave systems based possibly on multi-Gigahertz technology. Terrestrial cellular systems or (possibly) satellites in low Earth orbit.

The Network Interfaces are specific to each access system; they terminate each access network at physical, electrical, and network signaling levels. Some NIs incorporate special equipment (e.g., antennas, transceivers) needed on customer premises; some may be split between internal and outdoors chassis’ for environmental reasons.

The Access Gateway Manager commands Access Gateways (including Set Top versions) but does intervene with the low-level signaling, processing, or formatting up to the network layer, that goes on inside of them. Some Access Gateways will carry multiple classes of service and connections on one medium, or carry high bandwidth broadcast traffic; the AGM directs switching of inbound and outbound traffic.

- Controls several different Access Gateways simultaneously, if necessary. May operate multiple data streams in a single AG portal, and multiple AGs connected to the same access system.

- Controls calls and data exchanges in progress. Initiates signaling exchanges with access networks to support call forwarding and related functions, as instructed by the interface manager and applications. Sends and receives directory and related information requests.

- May coordinate cross-media services. For example, a packet radio control uplink to a network server that delivers IP format data via a unidirectional cable (data) channel.

- Relays tuning choices, interactive TV responses, and other commands received from user input devices (through the IB subsystem) after converting them into the appropriate form. Routes them to and from AGs through appropriate communication linkages. Some AGs (e.g., video boxes) may tune at the network edge to limit bandwidth needs.

The AG Manager implicitly commands invocation of low-level processing and signaling functions:

- low level signal processing streams in the AGs. May allocate internal pools of tuners, modems, codecs, decrypters that process and convert incoming or outgoing data streams.

- format conversions to/from internal or access systems’ communication protocols and signal formats.

Access gateway platforms may carry on-board service provider applications that should be distinguished from HNOS organic functions and applications, viz.:  

- Customer care, service administration and billing, with reporting and updating via network-initiated commands, and with consumer interaction and home-based data collection.

- Service provider GUIs and electronic program guides.

The Access Manager shall make these accessible with as much ease as other applications.
3. Essential Characteristics of an HNOS

The following are characteristics that the HNOS must have in order to support the wide range of functions outlined above. Most notably, the HNOS shall be able to transparently support small appliances and computing devices that cover a broad range of communication and computing capabilities. Older analog devices may interface into the HNOS through controllers: for example, via a gateway card residing in an RG.

The HNOS architecture shall support the eventual migration of network services to distributed applications, as service providers rely increasingly on the intelligence of the end nodes, such as set top boxes, network computers and PCs. Services will become distributed applications across the service provider networks as well as across clients and servers within homes.

The home network operating system shall address network heterogeneity-related issues through software abstraction, by providing:

- Operating system kernel software that makes minimal memory and hardware demands. The HNOS thereby becomes a tool that can integrate diverse platforms (appliances): smart and dumb, big and small, running as the native operating system on thin clients (e.g., on small appliances) or hosted under another OS (such as WINTEL). The kernel shall be available for many platforms

- End to end protocols and brokering support for all of the common networks.

- An end to end programming environment, with ability to have programs migrate between platforms using interpretive or fast compilation, and a common API and support. This includes APIs that support common programming languages for “Applets” such as JAVA and LIMBO, and also support for data streams that carry content labels (implying a format and content registry, with some middleware that interprets a small, agreed upon set of types).

- Support in middleware for common home control languages, such as CEBUS and X10. Support for common user interfaces with the HNOS, including DMTF, universal remote controls, and graphic user interfaces for VGA and NTSC TV monitors.

- Support for privacy control, and for secure communications in electronic commerce contexts. Secure resource sharing for distributed elements within and outside the home.

The essential design elements underlying HNOS shall be:

- **Small Size**: Complete instances of the HNOS, including basic applications, should be able to run comfortably in one megabyte of memory, which is modest by today’s standards. Subsets of the HNOS with severely curtailed functionality may also be available for use on very low cost appliance platforms. An advantageous way to accomplish this is to represent resources as files, thereby reducing the amount of interface code required in the HNOS kernel and in applications.

- **Portable OS**: The HNOS kernel and device drivers, the emulation kernel and the programming language interpreter shall be compact and able to run in native mode on very small or larger platforms, or in a hosted mode on common industry operating systems for PCs and set-top boxes. The interpreter or fast compiler and other performance-sensitive resources will be advantageously written in ANSI C to promote high portability. The remainder of the HNOS shall be written in a portable language (such as Java or Limbo) that uses the API and middleware resources conventionally offered across HNOS platforms to ensure portability.

- **Secure**: The HNOS should be able to authenticate connections between systems, advantageously using public key encryption algorithms in which both client and server possess public/private key pairs. Messages signed with a private key can be verified with the public key. In addition to authentication, the HNOS shall also provide
message digesting. The API provided by the HNOS shall provide a generic way to implement security in all applications that need to make use of it.

- **Network independent**: A standard communication protocol should allow applications and communications to be independent of the physical networks (aside from performance criteria). It shall not be necessary to design and develop local and remote proxy interfaces to implement inter-machine communication.

- **Real time performance**: control and communication functions demand predictable latencies for effective device control and handshaking, and for acceptable isochronous communication.

- **Scaleable**: The HNOS and home system components shall support scalability; that is, there should be no in-built limits in typical HNOS implementations on the number of high bandwidth access networks that can be connected, or on bus architectures of gateway devices. For high speed analog or digital sources (e.g., video), bandwidth may become contentious when multiple sources are present, and the HNOS shall potentially have to support content selection displaced to an edge RG (e.g., remotely commanded set-top box analogs used as servers feeding a home video network that cannot broadcast all the incoming material). Scalability also implies no inherent limits on the appliance population.

---

**Figure 2.2: End-to-End Software Architecture**
4. HNOS System Architecture

4.1. An NOS Definition

In one definition (Tanenbaum\textsuperscript{1,2}) of a network operating system, the NOS must provide a one-computer image of the network while permitting a high degree of autonomy to each computing node. It is a layer of functional capabilities residing above operating system and low-level networking software used by distributed applications to enable inter-operation and resource sharing. This kind of network operating systems is sometimes referred to as \textit{middleware}.

Middleware eases the development of distributed applications, but it offers no assurance of portability to different platforms. Further, on each particular machine, the middleware needs to be supported by an operating system, which makes it impossible for middleware to run on some rudimentary information appliances such as set-up boxes. Continual research on removing these limitations led to the advent of a \textit{full network operating system}\textsuperscript{3}.

Running either standalone on inexpensive hardware or as an application under conventional operating systems, a \textit{full network operating system} provides a complete environment for portable distributed applications through multiple levels of abstraction:

- A \textbf{virtual machine (VM)}, which serves as the common execution environment for portable applications. Applications built on the full network operating system VM are compiled onto its instruction set, which ensures portability across hardware platforms. Further, the instruction set of the virtual machine shall match well with those of the modern physical machines to ensure efficient execution.

- A \textbf{virtual operating system}, which serves as both an extended machine and a resource manager. In the former role, it provides an application programming interface (API) shielding the application developer from the low level hardware and network, and from complexities of dealing with, for example, inter-process communications, networking, and security. In the latter role, it performs the actual task of resource management, including transparent, secure sharing of resources among processes throughout the network.

- A \textbf{virtual network interface}, which provides a common network interface to the applications. Applications therefore can be developed independent of the underlying network or transport.

These principles shall be applied to the HNOS.

4.2. Technical Approach for an HNOS

The HNOS shall be a distributed operating system that can incorporate clients and servers with a wide range of characteristics. The design shall include the following three basic principles:

- \textbf{Resource abstraction as files}: System resources are represented as files in an hierarchical file system.

- \textbf{Name spaces}: The application view of the network is a single, coherent name space that appears as a hierarchical file system but may represent physically separated resources.


• **Standard communications protocol**: A standard, lightweight communications protocol is used to access all resources, both remote and local.

### 4.2.1. Resources Abstracted as Files

The HNOS interfaces to resources shall in general be represented as a set of dynamic files that an application can manipulate as required. File systems will be used to represent storage devices, processes, services, networks, and network connections, interfaces to networks. The file-oriented interface to resources involves resolving names in a hierarchical tree, attaching to them by name, and accessing their contents by read and write calls.

The use of files as a central concept in the HNOS has the following advantages:

- File interfaces are simple and well understood across a wide variety of operating systems. There are generally a small set of well-defined operations, such as open, read and write.
- The amount of interface code needed is small, keeping the HNOS small, reliable and highly portable.
- Naming conventions for files are well known, uniform, and easily understood.
- Access rights and permissions to files are simple, yet can be used to ensure multiple levels of security.

File names and contents can be dynamic and can be synthesized on per-demand and on a per-client basis. For example, the HNOS would create a dynamic set of files to represent each active TCP connection. An application then reads from and writes to these files while the target TCP connection is still alive. When the connection is no longer necessary, the files are closed and the connection is automatically closed as well.

The HNOS structure is therefore implicitly constrained by fixing the allowable set of file methods. File servers can easily broker file operations such as granting access. The file structure also implicitly defines the mechanisms for garbage collection and accounting.

### 4.2.2. Name Spaces

A second key HNOS principle is the computable name space, by which an application builds a unique private view of the resources and services that it needs to access. Each set of resources and services is represented as a hierarchy of files and is accessible via the familiar file access operations. The various resources and services being used by a process are combined into a single rooted hierarchy of file names - its **name space**.

The resources accessible to an individual name space can be located on a single client or on multiple servers throughout the network. File systems provided by different servers can be combined into a single, unique name space, which becomes the application’s own view of the network.

### 4.2.3. Standard Communications Protocol

The third key HNOS element is the use of a simple, standard communications protocol. This protocol will represent network transports, network devices, and network connections as file systems.

The protocol supports remote access to files and allows a remote machine to use these interfaces as gateways. Using the same set of files to represent different devices allows the creation of common tools to serve several networks. Network connections represented by these files behave the same way for all networks and thus allow applications to contain no network-specific code.
5. HNOS Operation and Components

The HNOS shall provide transportable, lightweight versions of the following four primary components:

- An OS kernel with native mode or hosted implementations.
- An small, transportable programming language with on-the-fly compilation or interpretation. It will be used for low- to mid-level components of the OS (enhancing its portability) and also for applications, some of which will be migratory among home- or network-based nodes sharing the HNOS API.
- A virtual machine (VM) with a platform-independent API that has many native mode and hosted implementations.
- A small communications protocol that supports transparent networking.

Figure 5.1 shows these elements in a view of the HNOS architecture. Each provides distinct functions, integrated together to support the HNOS functionality layer by layer:

- At the Application Layer, applications written in the HNOS programming language run on top of the virtual machine API, which interprets instructions and invokes the appropriate system calls that are necessary to complete the requested actions.

The virtual machine may be present on and shall be capable of running applications on either clients or servers, provided applications are written in the HNOS programming language. The APIs (interfaces to lower levels of
HNOS functionality) are the same for both clients and servers. The API provides access to system resources by including core modules that contain explicit interface specifications.

- **The Kernel OS Layer**, below the application layer, provides operating system functionality, including:
  - Namespace management
  - Security
  - Database Support
  - Graphics functions
  - Process management / scheduler
  - Memory management

These elements behave *identically* whether an instance of the kernel is running directly on hardware as the native operating system or as an emulator on a host operating system. Although each environment needs an individual kernel implementation, they are small and are also largely written in the HNOS’s programming language, enhancing the portability.

- **The Physical Layer** consisting primarily of physical components of the system, such as devices and networks. The exception is when the HNOS is running in the emulation environment, where access to the physical layer is via the host operating system.

### 5.1. The OS Kernel

The HNOS kernel provides mechanisms for data and resource management, networking, and security. Its purpose is to provide an environment for running applications under the HNOS virtual machine. The kernel provides the following services to the VM:

- Process creation
- Namespace management
- Data streaming
- Network protocols

The kernel is also responsible for I/O and hardware connections via device drivers. The kernel shall be compact and sparse, resulting in an operating system that is small in size and thus is easier to port to new platforms and architectures.

#### 5.1.1. Native vs. Emulation Kernels

The goal of the emulation kernel is to hide the differences between host operating systems.

For purposes of application development, the emulation kernel is identical to a native kernel. When a hosted, or emulation, kernel is running, the environment is the same as in native mode, although a host operating system may be providing many of the services that a native kernel provides. The HNOS kernel needs to efficiently perform resource management that depends on devices on the host system, allowing it to interact with existing operating systems without modification to the upper, application layer.
Various native and hosted implementations of the kernel need differ from each other only in a small proportion of hardware- or operating system-dependent code, provided that much of the kernel is written in its own transportable programming language. Examples of OS’s written in this structure are UNIX/C, INFERNO/LIMBO, and some older threaded code interpretive languages (e.g., FORTH).

Both hosted and native modes can share shell-like interpreters and lightweight windowed GUI’s, providing a system control console and a means for launching applications.

5.1.2. Memory Management

Memory management shall be effective with small non-disk systems, since virtual memory will typically not be available. Performance should degrade gracefully when memory constraints are reached. A two level allocation mechanism may be desirable, in which one manager element controls large contiguous memory blocks, while a second level divides blocks into small pools that applications control when resources become scarce. The pools needed will include: network buffers, general (application) memory structures, a heap for the VM, graphics and font memory files.

5.1.3. Scheduling

The kernel shall provide preemptive scheduling of processes that are responsible for managing and servicing protocol stacks, media copies, alarms, interrupts, and the like. The scheduler shall maintain multiple priority classes and run queues. The highest priorities serve isochronous communications and true real-time tasks such as voice and video telephony.

5.1.4. Devices

Each device is represented by a file tree that can be attached to a name space for use by applications, and is implemented by a device driver. Device-specific behavior is obtained by responding to open, read, and write system calls. The OS kernel may also maintain a table of virtual devices that support communication protocols.

5.1.5. Name Space Construction

A namespace provides a customized, local view of resources available through the network to a user and application. It can be built up by a two step process of mounting and binding. A local namespace (looking like a directory tree) can be extended by grafting on subtrees from “servers” accessible to the HNOS, thereby gaining access to resources available to those (perhaps virtual) machines. Once incorporated into a local namespace, remote resources become indistinguishable from local ones.

- **Mounting** a remote “servers” namespace component to a local namespace (file tree) adds the remote subtree to the local one so that its members can be accessed in the same way as local files. Home applications will typically pre-mount namespaces through system initialization and when they are invoked using scripts. Other mount invocations allow applications to construct unions of directories that concatenate component subtrees so that they appear to be one, with a specified order.

- **Binding** maps one file name in the namespace to another. It appears to duplicate an existing name space subtree at some point in another name space. That is, the bind call places a mounted name space component in a desired location on an existing directory structure.

The effects of binding and mounting are undone by an **unmount** call.
Existing systems, such as UNIX, manage namespaces through tables that translate user file descriptors into ‘channel structures’. File descriptors are passed as parameters in a system call, then converted to a channel structure to provide the kernel with a representation of that file object. The kernel represents a name space as a set of relations between channels in an internal mount table that stores a list of bindings between channels.

5.1.6. Security

Key security features and drivers shall be built into the HNOS so that they cannot “spoofed” or replaced by other routines. Namespaces can also help restrict access to resources, making available only those that each application actually needs. Cryptographic features shall be built into security routines and the API, enabling the following security provisions

- **Mutual Authentication**: allowing two users or applications that with to communicate to establish that they are who they say they are, using a protocol and keys.
- **Message Digesting**: ensuring that an interloper can not modify messages sent between users. This is especially important in financial transactions.
- **Digital Signatures**: combining the above to provide undeniable transaction control - testifying to the identity of a message sender in a way that is at least as hard to forge as a conventional signature.
- **Encryption**: protecting the confidentiality of the contents of messages so that only the parties for whom they are intended can decrypt and read them.

5.1.7. Database Support

Given the complexity of Home environments, the HNOS requires database support for its functional subsystems (described earlier) to function properly. The HNOS shall provide a programming interface (available to middleware) that permits standardized access to home-resident or networked databases. It has two elements:

- A Virtual Machine module that that provides an API for databases compatible with a generic SQL (a common database language called Structured Query Language). It offers programs written in the HNOS procedural language with access to powerful database management for internal and external use.
- A “server” module, which is a software interface that either supports access to a physical server that works under the HNOS database language and namespace or adapts external devices to the HNOS (a virtual server adapter).

5.2. Programming Language

The HNOS shall provide one or several general-purpose programming languages, that will be used to create portable applications, gateways and interfaces to non-HNOS environments, and segments of the kernel itself. The OS Kernel shall provide interpretation and/or on-the-fly compilation. Advantageously, at least one of these will have a familiar C-like syntax, be strongly typed, provide automatic garbage collection, and support only very restricted pointers to protect against viral or innocent memory incursions.

It is important that at least one supported programming language have broad acceptability in the user community.
At run time, applications ("Applets") may migrate to an HNOS VM instance, experience just-in-time compilation at the discretion of the programmer, or alternatively be interpreted by the VM. Applications run on any platform that supports the API. Compiled code shall have a hardware-independent representation (or byte code) for execution on the virtual machine.

### 5.3. Universal Virtual Machine

The "universal" virtual machine shall provide an execution environment for one or several programming languages (for example, JAVA + LIMBO + dialects). It shall control multiple threads (control hierarchies each created by a spawn statement) and multiple OS processes (host OS processes when running under another OS). There are separate schedulers for operating system processes and threads. Each new emulation environment requires file system interface programs that connect to local device drivers and to the file system used by the host operating system. A candidate for the VM must present programs with abstract solutions for:

- **Memory Organization**: For example, a single VM instance (all modules and threads within it) may control the heap (no sharing).
- **Garbage Collection**, to restore resources to availability immediately after use, using an independent background process running behind the VM.
- **Scheduling / program execution**: Multiple threads are placed onto a run queue for execution. Threads are multiplexed onto processes. A given thread may be executed by several processes in sequence before it completes, and multiple threads may be attached to a single process.

### 5.4. Transparent Networking

One of the HNOS' goals is to encourage resource sharing in a distributed environment, with participation by a wide variety of hardware platforms. To facilitate this, each platform shall present the same view of the network to the user.

A home network configuration may be made up of many distributed clients and servers, anywhere, additionally connected to each other by a variety of physical media and network protocols.

#### 5.4.1. Communications Protocol

Like other HNOS services, network transports, network devices and network connections are represented as file systems, and abstracted as HNOS "servers" that provide access to one or more hierarchical file systems on request from a client, usually through a bi-directional connection that may be "opened" or "closed" on demand.

Using the same set of files to represent different devices allows the creation of common tools that can serve several networks or interfaces. Some familiar services whose implementation on other operating systems is more complex may be provided just by echoing the contents of the interface files. Network connections represented by these files behave the same way for all networks and allow application programs to contain no network-specific code.

The HNOS protocol shall run on top of any common, reliable transport level protocol, counting on reliable, sequential delivery. For protocols not meeting these requirements, HNOS shall provide mechanisms to order the messages.

The HNOS shall in particular support the Internet Protocol (IP) suite, including Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).
File operations between any two machines are accomplished by sending requests and receiving replies via the HNOS protocol in a manner similar to remote procedure calls. A file operation indirectly creates messages that depend on the call, but the complexity is concealed from the developer, who simply invokes mount, open, read and similar system commands as necessary.

The protocol contains messages that perform:

- Navigation over the file hierarchy
- Control of file attributes
- Access to data in files
- Miscellaneous functions

### 5.4.2. Hiding Network Complexity

The HNOS hides network complexity from applications.

- An application needing access to services that reside on remote servers shall use a few simple operations to incorporate them into a single, locally-represented name space that may encompass multiple remote machines and multiple networks, appearing to the client application as a local file system.

- The HNOS communications protocol unifies different parts of the local name space that may be constructed from the vast collection of files on the network. File operations between any two machines are accomplished by sending requests and receiving replies via the common protocol messages.

Application programs generate messages only indirectly by seemingly local file system operations (open, read,..).