İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

HOME AREA NETWORKING WITH UNIVERSAL PLUG AND PLAY TECHNOLOGY

M.Sc. Thesis by
Nuri EnderKARAGÖZ

Department : Electronics and Telecommunication Engineering

Programme : Telecommunication Engineering

JUNE 2007

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

HOME AREA NETWORKING WITH UNIVERSAL PLUG AND PLAY TECHNOLOGY

M.Sc. Thesis by
Nuri Ender KARAGÖZ

(504051341)

Date of submission : 7 May 2007

Date of defence examination : 13 June 2007

Supervisor (Chairman): Prof. Dr. Mine KALKAN

Members of the Examining Committee Prof. Dr. Melih Pazarcı

Doç. Dr. Selçuk Paker

Yrd. Doç. Dr. D. Turgay Altılar

JUNE 2007

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

EVRENSEL TAK VE ÇALIŞTIR TEKNOLOJİSİNİ İÇEREN EV YEREL AĞLARI

YÜKSEK LİSANS TEZİ

Nuri Ender KARAGÖZ

(504051341)

Tezin Enstitüye Verildiği Tarih : 07 Mayıs 2007
Tezin Savunulduğu Tarih : 13 Haziran 2007

Tez Danışmanı :	Prof.Dr. Mine KALKAN
Diğer Jüri Üyeleri	Prof. Dr. Melih Pazarcı
	Doç. Dr. Selçuk Paker
	Yrd. Doç. Dr. D. Turgay Altılar

HAZİRAN 2007

PREFACE

I would like to thank to Prof.Dr.Mine KALKAN for all her help and guidance during the preperation of this thesis. I would also like to thank to all my friends and colleagues for their valuable supports. Many special thanks to my wife, my son and my parents for their great patience, motivating and making my life easier attitudes during that period.

June, 2007 **Nuri Ender KARAGÖZ**

TABLE OF CONTENTS

ABBREVIATIONS

ACL	: Access Control List
AES	: Advanced Encryption Standard
ARP	: Address Resolution Protocol
A/V	: Audio/Video
AVT	: Audio/Video Transport
CCK	: Complementary Code Keying
CDS	: Content Directory Service
CMS	: Connection Manager Service
CP	: Control Point
CSMA/CA	: Carrier Sense Multiple Access with Collision Avoidance
DCP	: Device Conrol Protocol
DECT	: Digital Enhanced Cordless Telecommunications
DHCP	: Dynamic Host Configuration Protocol
DNS	: Domain Name Service
DPWS	: Devices Profile for Web Services
DSL	: Digital Subscriber Line
DSSS	: Direct Sequence Spread Spectrum
EPF	: Electronic Picture Frame
FHSS	: Frequency Hopping Spread Spectrum
GENA	: General Event Notification Architecture
HAN	: Home Area Network
HAVI	: Home Audio Video Interoperability
HDTV	: High-definition Television
HMAC	: Keyed-Hash Message Authentication Code
HTML	: Hypertext Markup Language
HTTP	: Hypertext Transfer Protocol
HTTPMU	: Hypertext Transfer Protocol Multicast
HVAC	: Radio Detection and Ranging
IEEE	: Institute of Electrical and Electronics Engineers
IP	: Synthetic Aperture Radar
IrDA	: Infrared Data Association
MIMO	: Multiple Input, Multiple Output
NT	: Notification Type
NTS	: Notification Subtype
OFDM	: Orthogonal Frequency Division Multiplexing
OSGI	: Open System Gateway Architecture
PDA	: Personal Digital Assistant
PHY	: Physical Layer Of OSI Model
PLC	: Power Line Communication
PNA	: Home Phoneline Networking Alliance
RCS	: Rendering Control Service
SC	: Security Console
SCP	: Simple Control Protocol
SDTV	: Standart Definition Television
SHA	: Secure Hash Algorithm
SID	: Security Identifier

SMTP	: Simple Mail Transfer Protocol
SOAP	: Simple Object Access Protocol
SSDP	: Simple Service Discovery Protocol
SSL	: Secure Sockets Layer
SUPP	: Simple UPnP Proxy Protocol
TCP	: Transfer Control Protocol
TDMA	: Time Division Multiple Access
UDP	: User Datagram Protocol
UML	: Unified Modeling Language
UPnP	: Universal Plug and Play
URI	: Uniform Resource Identifier
URL	: Uniform Resource Locator
USB	: Universal Serial Bus
UTF	: Unicode Transformation Format
UTP	: Unshielded Twisted Pair
UUID	: Universally Unique Identifier
UWB	: Ultrawide Band
VCR	: Video Cassette Recorder
WLAN	: Wireless Local Area Network
WMAN	: Wireless Metropolitan Area Networks
VoIP	: Voice Over Internet Protocol
WPAN	: Wireless Personal Area Networks
WS	: Web Services
WSDL	: Web Services Definition Language
WS-I	: Web Services Interoperability
W3C	: World Wide Web Consortium
XML	: Extensible Markup Language

TABLE LIST

FIGURE LIST

EVRENSEL TAK VE ÇALIŞTIR TEKNOLOJİSİNİ İÇEREN EV YEREL AĞLARI

ÖZET

Bu çalışma, ev içerisindeki kablolu ya da kablosuz akıllı cihazların, farklı özellikteki bilgisayarların birbirlerine otomatik olarak bağlanması ve birlikte çalışması, ev yerel ağının işletim sisteminden, programlama dilinden, iletişim mekanizmalarından ve fiziksel ortamdan bağımsız olabilmesi için uçtan uca dağınık ağ mimari yapısına ve TCP/IP, HTTP, XML gibi farklı İnternet standartları ve protokollerine imkan tanıyan evrensel tak ve çalıştır teknolojisini inceler. Bu teknoloji, sıfır konfigürasyonlu ağ oluşturma ve otomatik cihaz keşfini sağlayarak ağ oluşturmayı kullanıcıların yapabilecekleri basit yapıya dönüştürür.

Ayrıca, bu çalışma isteğe bağlı transfer protokollerini ve içerik formatlarını destekleyen, ses ve görüntü içeriğinin kontrol noktası tarafından herhangi bir müdaheleye gerek olmadan cihazlar arası akmasını, kontrol noktalarının herhangi bir cihaz tipinden, içerik formatından, transfer protokolünden bağımsız olmasını sağlayan, ortam çeviricisinin yakınındaki ortam sunucusunun servislerini keşfedip gerekli ayarlamaları otomatik olarak yaparak kullanıcının ev eğlence kullanımı senaryosuna uygun olarak favori parçasını seçmesine olanak sağlayıp kullanıcıdan herhangi bir müdahele gerektirmeyen geniş kapsamlı evrensel tak ve çalıştır ses ve görüntü mimarisini tanımlar.

Uygulama safhası, kullanıcı müdahelesi olmadan ortam içeriğini depolayan ve akışı başlatan ortam sunucusu ile içeriği ortam formatından tamamen bağımsız olarak çalınmaya uygun hale getiren ortam çeviricisi arasında, sıradan bir TV ya da müzik setine evrensel tak ve çalıştır imkan ve kabiliyeti kazandırabilmek için donanımsal arabirimi ortam çeviricisi olarak oluşturarak ve bilgisayar ya da dizüstü bilgisayarı kontrol noktası ve ortam sunumcusu olarak programlayarak, ses ve görüntü akışını sağlar.

HOME AREA NETWORKING WITH UNIVERSAL PLUG AND PLAY

SUMMARY

This study provides an overview of Universal Plug and Play (UPnP) Technology, which allows an architecture for pervasive peer-to-peer network connectivity of intelligent devices, PCs of all form factors, and wireless devices within the home, and builds on Internet standards and technologies, such as TCP/IP, HTTP, and XML, to enable these devices automatically connect with one another and work together, to build home networking independent of any particular operating system, programming language, transport mechanisms, or physical medium. UPnP Technology is all about making home networking simple and affordable for users, supporting zero-configuration networking and automatic discovery,

Additionally, the study defines overall UPnP A/V Architecture which supports arbitrary transfer protocols and content formats, enables the A/V content to flow directly between devices without any intervention from the Control Point (CP), makes CPs to remain independent of any particular device type, content format and transfer protocol and requires no assistance at all from the user since a Media Renderer (e.g. mp3 player) can discover available services from a Media Server in its neighborhood, perform the necessary setup automatically, and finally enable the user to select and play their favorite song in a home entertainment usage scenario.

The implementation phase provides Audio/ Video Transport (streaming) without any user intervention between Media server, and Media Renderer by building and programming a hardware interface as Media Renderer, in order to make an ordinary A/V device UPnP enabled, and programming a laptop as a CP and Media Server.

1. INTRODUCTION

The digital age of consumer electronics is bringing with it faster computing at lower costs. The Internet revolution and distribution of broadband access to different digital consumer electronics and their interoperation introduces a new wave of technology into the home. The rapid proliferation of personal computers (PCs), the Internet in homes, advancement in telecommunications technologies, progress in the development, decreasing cost of smart devices, that allow users to control and monitor events in consumer-based appliances, and consumer demand for content rich, have increasingly emphasized the need for an in home area networking. Furthermore, as these growth and advancement trends continue, the need for simple, flexible, and reliable home networks will greatly increase.

Home Area Networks (HANs) represent a new application of technology aimed at supporting communication between heterogeneous devices in the home which cover the range from the well-known PCs, laptops but also mobile computing devices, Internet appliances and home sensors [1]. The goal of HANs is to leverage the individual capabilities of these devices by integrating them into a larger more powerful network. This integration will lead to a paradigm shift in the way people use and manage the electronic and computing devices at home and use content and communications to enhance their living experience.

HANs facilitate communication among appliances, home systems, entertainment products and information devices in a home so that they can work cooperatively and share information. A device connected to a HAN gains the capabilities of other networked devices, and as a result the device can provide a service or function that it would have otherwise been incapable of providing alone.

Local area networks (LANs) and Wide Area Networks (WANs) have become ubiquitous. The network hierarchy has been rapidly moving lower in the chain towards smaller and more personal devices. The number of networked homes, as well as the number of networked devices within the home, are proliferating at an accelerated pace.

In the second section of this thesis, the architecture and the systems of home area networking, different home networking choices evolving solutions which contain structured, new wiring and wireless technologies, are handled and overviewed [2].

All sorts of devices (PCs, mobile phones, cameras, handheld computers and so on) are increasingly connecting to networks, and they are using a multitude of connectivity methods to do so. This increases the need for self-configuring networks that allow devices to easily and automatically join and leave networks and to learn about other connected devices. Home networks, automotive networks and similar environments demand new technologies [3] that can automate device and service discovery and control, obviating the need to administer these networks.

Universal Plug and Play (UPnP) technology [15] is a unifying force. It defines a framework for a common, network-based platform for device collaboration. It implicitly enables the notion of a digital home platform which means an abstraction that brings coherence to previously separate islands of connectivity in the home. The digital home platform will span wired and wireless networks, entertainment devices, telephone equipment, home control, and so on, linking the various networks in the home (entertainment, home control and automation, communications, and the data network) into a single logical network of programmable devices, as shown in Figure 1.1.

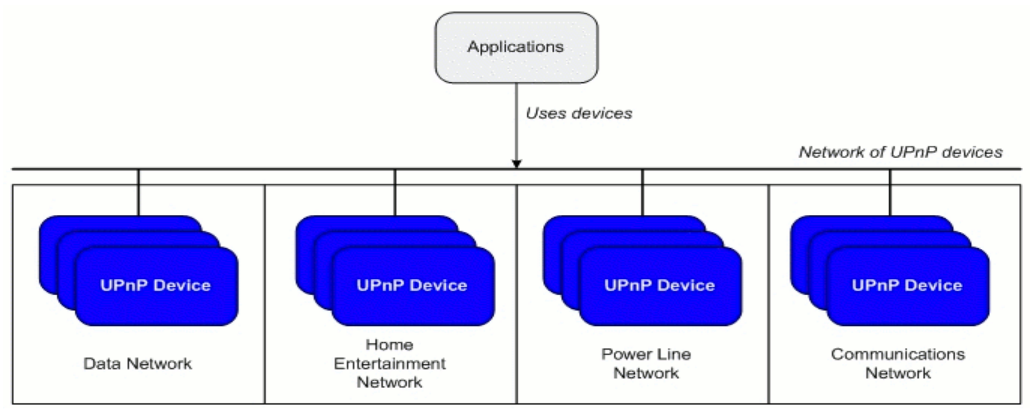


Figure 1.1: UPnP: A Unifying Network Technology

The resulting logical network will contain a multitude of UPnP devices that will be able to interact with each other. Control Points (CPs) on the network, such as the PC in the den, or the television and remote in the entertainment area, will have programmatic access to all of the devices on this network and will be able to run programs that aggregate the devices into innovative applications. Thus, the UPnP foundation of the digital home enables a transition to new home networking usage models involving higher levels of automation, easier integration of devices from different manufacturers, and hides much of the complexity of installing, configuring and using networked devices in the home from users. This transition will have great benefits for people in the home, as the focus will shift from a specific user interaction model to devices supporting user activities in the home in their natural settings.

The third section desribes the UPnP technology which includes UPnP protocol stack, layer, and UPnP specific protocols, UPnP device architecture [20] and UPnP technology's relation to other technologies which are alternate standards for home networking subjects are inspected.

Security is an important issue in all device and network types. UPnP Security is concerned with networks in which more than the user's own CPs are present on the physical network and able to reach the user's devices with control messages. It is important in such cases that devices and CPs employ security mechanisms to verify the identities of each other and provide basic levels of access control for resources on the network.

UPnP Security has defined a combination Device and CP called the Security Console (SC). Its purpose is to take security ownership of Devices and then to authorize CPs (or other SCs) to have access to Devices over which the SC has control. An SC can own a Device, meaning that it has the right to edit that Device's Access Control List or it can be given some subset of rights to the Device and have the privilege to grant all or some of those rights to another SC or CPs (by way of an authorization certificate). The SC also has the job of defining names of individual CPs and of groups of CPs.

The Security architectute of UPnP Technology [29] including UPnP Security actors, Discovery and Component Naming, security Ownership, Access Control and the Logic of Operation is overviewed in the fourth section.

UPnP 1.1 device architecture was created for backward compatiblity with UPnP 1.0. Since UPnP 2.0 was not backward compatible and was also designed around still evolving Web Services drafts. There are several improvements in the UPnP 1.1 device architecture, mostly resulting in improved performance and improved compliance with referenced standards. Now, one stack can interoperate with the UPnP 1.0 device architecture, UPnP 1.1 device architecture and web services.

In the fifth section, improvements in UPnP Architecture including Improved IPv6 support, Discovery improvements, HTTP 1.1 Profile and Deprecation of HTTP Extension Framework, support for full XML-schema data types for Description, SOAP

(Simple Object Access Protocol) 1.1 Interoperability Profile and WS-I (Web Services Interoperability) Basic Profile Compatibility, Web Services and Device Profile for Web Services are inspected [30].

UPnP A/V (Audio, video) Architecture [44] defines the general interaction between UPnP A/V Devices and associated CPs. It is independent of any particular device type, content format, and transfer protocol. It supports a variety of devices such as TVs, VCRs, CD/DVD players/jukeboxes, set-top boxes, stereos systems, MP3 players, still-image cameras, camcorders, electronic picture frames (EPFs), and the PC. The A/V Architecture also allows various types of transfer protocols such as HTTP and Real-time Transport Protocol (RTP). The sixth section describes the A/V Architecture including Media Server, Media Renderer and CP.

In seventh section, a CP, a Media Server and a Media Renderer has been designed, coded and implemented. The implementation is based on making a device UPnP enabled that is originally a non-UPnP device. That is a television has been made UPnP technology enabled device as a Media Renderer. Since UPnP technology is independent of physical layer and operating system (OS), a wireless technology (802.11 b/g) has been chosen for physical communication of Devices and CP, and Unix OS for Media Renderer and Windows XP OS for Media Server and CP applications. By doing that, UPnP A/V Architecture is implemented in accordance with the overall UPnP technology.

2. HOME AREA NETWORKING

The HAN improves communication and allows the sharing of expensive resources, such as sharing a printer among many desktop computers or sharing a common high-speed Internet connection among multiple computers. More traditional computing devices are being interconnected in the home. Even commonplace home appliances, such as air conditioners, heaters, refrigerators, and lighting systems, are finding new useful services through connections to the Internet. These trends, coupled with the emergence of new classes of small, intelligent, network-ready personal computing devices, including smart phones and PDAs, are making the home an increasingly important networking center.

The HAN may be in the form of a wired network, using for example Ethernet as the data carrier and Cat5 cable as the physical layer, or a wireless network using, for example, the IEEE 802.11 protocol. The development of broadband modems, which allow greater data rates to the home, has enabled both higher speed and shared Internet access, and the downloading of files or the streaming of video, to become viable.

The wired HAN may use existing telephone extension wiring or use new cables fitted around the home. The wireless HAN, a more expensive but less obtrusive technology is easier to fit and removes the need for new wires and is an attractive option [9]. Figure 2.1 shows devices connected by both wired and wireless methods.

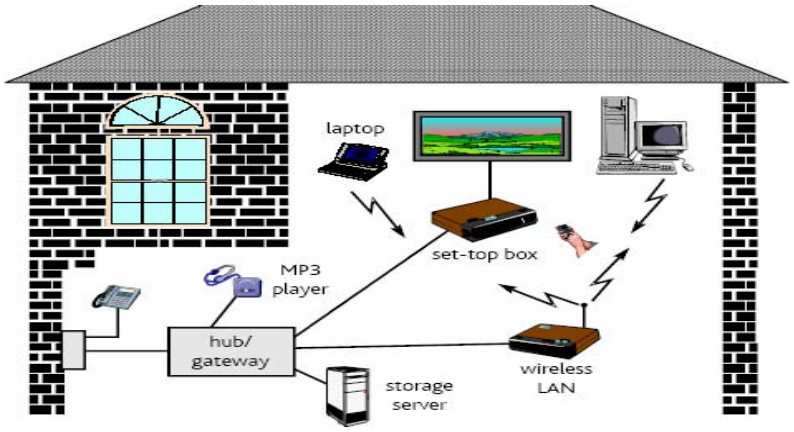


Figure 2.1: HAN Connects PC or Set-top-box Using Different Technologies within Home.

2.1 Architecture of Home Networking Systems:

Home networking has four aspects, which include broadband access, residential gateways, a vast range of information appliances, and the technologies that bind them all [5]. Interconnectivity or home networking technologies are shown in Figure 2.2.

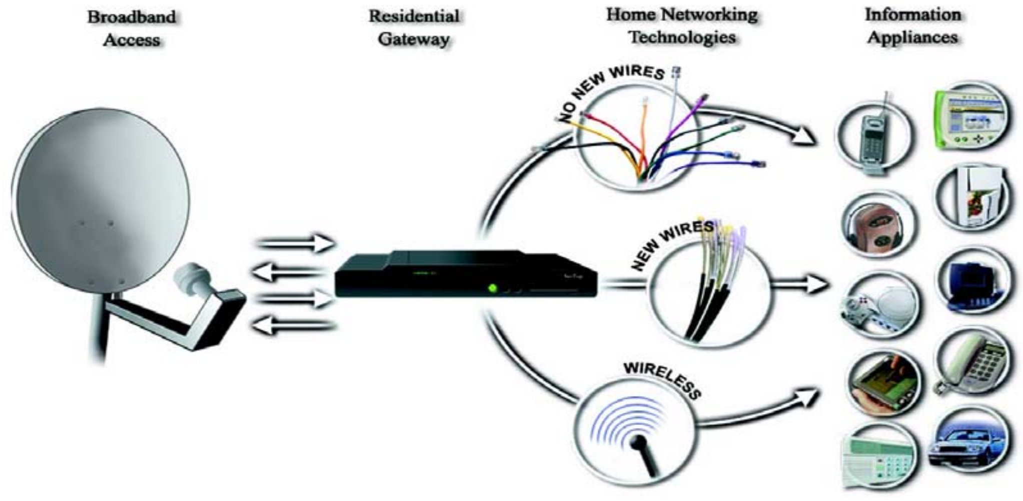


Figure 2.2: Architecture of a Home Network

Table 2.1: Home Networking Systems

	Market Requirements	Solutions Available
Broadband Access	High-speed access for data, voice, and video; simultaneous up-link and down-link communication; support simultaneous and multi-user access	xDSL, Cable, ISDN, Powerline, Satellite, Mobile/Wireless
Residential Gateway, **Home Gateway, or Services Gateway**	Provides access into the home;remote management access platform; bridging between different networks; firewall and security; e-services capabilities	Open-System Gateway Initiative (OSGI), Jini, UPnP, HA/VI
Home Networking Technologies	Low cost; speed; mobility; quality of service; reliability; ubiquity; ease-of-use	No New Wires (Phonelines, Powerlines), New Wires (Ethernet, optical fiber, IEEE 1394/ FireWire, USB 2.0); Wireless (Bluetooth, WLANs- IEEE 802.11)
Information Appliances	Digital electronics with advanced computational capabilities that add more value and convenience when networked	Digital TV/HDTV, set-top boxes, Internet screens, phones, digital VCRs, gaming consoles, MP3players, cordless phones, security systems, utility meters, PCs, Web pads, Web terminals, PDAs, digital cameras, auto PCs, etc.

2.1.1 Broadband Access

Digital broadband access to the home has revolutionized communication by providing a very fast, two-way communication channel. Internet access and the ability to transmit real-time audio and video anywhere at anytime is a revolutionary concept helping drive broadband access to the home. Rapid technological advances have immensely extended the list of possible broadband platforms that consumers can use to access the Internet. Solutions available include xDSL, cable, power line, satellite, and wireless platforms.

2.1.2 Media Gateways (Residential Gateways)

The HANs enable users to get information about the home's condition, to remotely control home systems and appliances, and to gain access to information and entertainment resources both from inside (such as a computer hard drive) and outside the home (for instance, from the Internet). To provide these benefits to home consumers, different devices of the home network must be able to communicate with each other to provide services despite their implementation differences. This requires the management and coordination of discovery methods that work across heterogeneous device technologies and complex home networking architectures. To achieve this management and coordination transparently without user intervention is a complex task which, until now has been the major factor responsible impeding the wide acceptance and delivery of advanced services into the home. A common solution is to exploit a central connection point often referred to as a residential gateway. The residential gateway acts as a "bridge" between the wide area network (e.g., Internet) and the home network. The gateway provides users access to home devices and control over the contents and leverages two technological trends namely, the ubiquity of broadband connectivity and Internet access in homes, offices, vehicles and mobile/portable devices, and the emergence of new applications and services. Starting as a low-cost home hub device, as different network technologies develop, residential gateway would evolve to become an intelligent gateway, and is required to manage the devices and content flow between devices, set up the devices and add protection and security. Instances of possible terminals in the home that may require networking are shown in Figure 2.3.

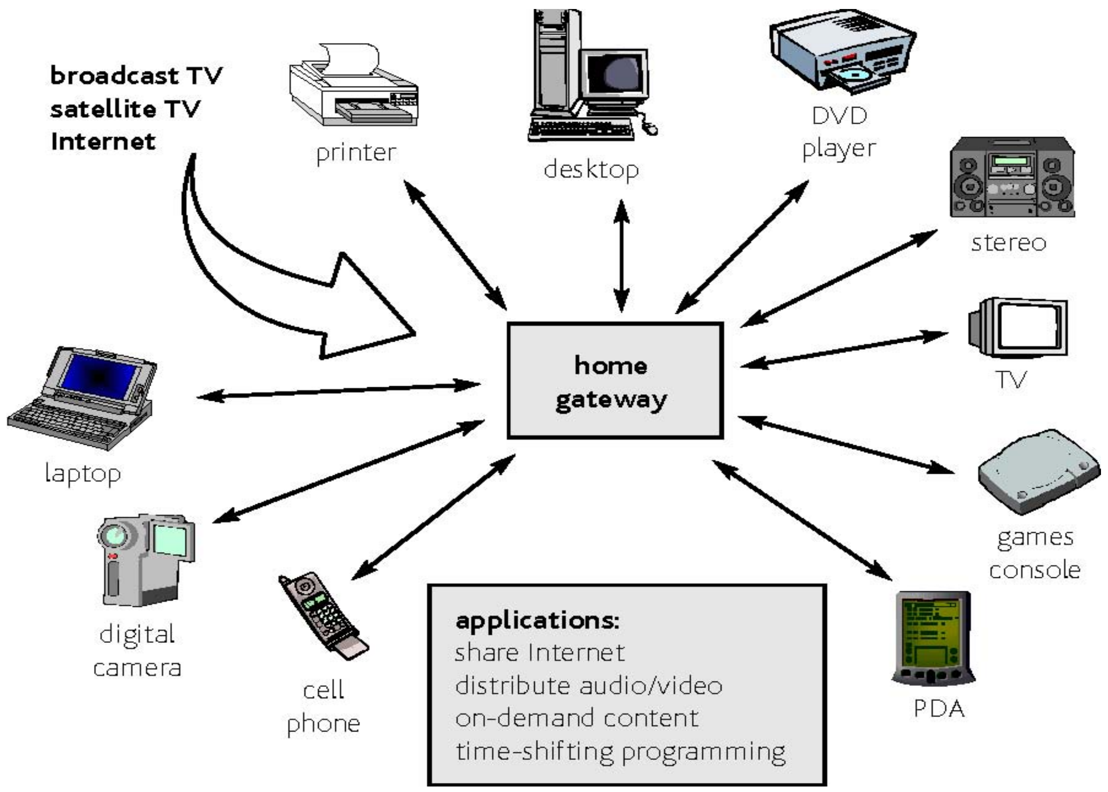


Figure 2.3: The Home Gateway

It needs to manage the devices, the data flowing between these devices, and the data flowing into the home.

2.1.3 Home Networking Technologies (Information Appliance Network)

The Information Appliance Network (IAN) consists of high-speed in-home data networking technologies that distribute Internet access to ubiquitous access points and appliances. The IAN provides interconnectivity for all home appliances. A wide variety of technologies exist for interconnecting devices within the home, but no single technology meets all of the requirements for the diversity of applications that will be created. While traditional Ethernet systems offer a robust and proven solution, most consumers do not have the time, interest, or knowledge to rewire their homes.

Fortunately, the emergence of “no new wiring” technologies offers possibilities for solving the mass-market home networking issue. New technologies include wireless, phone line, and power line solutions. Since each solution presents distinct benefits and drawbacks, many organizations are beginning to suggest that all of these technologies will co-exist in multi layered home network architecture. Types of home networking technologies involve existing wiring (no new wires), structured wiring (new wires) and wireless. Home networking technologies will be handled later in that section a little bit more detailed as home networking choices.

2.1.4 Information Appliances

These intelligent devices have the ability to communicate and interoperate using the IAN. In cases where the IAN is connected to a broadband access, these devices will be able to connect to the Internet. Networking these digital appliances will bring improved convenience and flexibility for the consumer. The functional requirements of digital appliances are ubiquity, reliability, cost, speed, Quality of Service (QoS), security, remote management and ease-of-use.

2.2 Motivations to Network a Household

The motivating factors that encourage people to network different classes of household appliances and computers together are;

- **Leverage existing investments**: First and foremost, people want to leverage their investments in expensive appliances such as computers, set-top boxes, personal video recorders, digital cameras, and cable modems. Hence, sharing hardware resources is the number one motivating factor for consumers investing in new home networking interconnection technologies.
- **Shared Internet access**: The second most popular motivating factor fueling the deployment of home networks is shared Internet access. Home networks, in conjunction with devices called media gateways, allow different members of a family to simultaneously use a single, fast Internet access, thus saving money.
- **Interconnecting subsystems**: Other motivating factors include the ability of a home networking infrastructure to interconnect different types of subsystems together. For example, home security systems are also defined as a network, but instead of interconnecting devices like printers and PCs, in-home security networks connect different types of sensors to a central controller. Integrating this type of network with an existing PC-based home network helps people expand the functionality of their security system and better manage different subsystems.
- **Rise of multi-PC households**: The rapid growth of multi-PC homes indicates that the number of nodes in a PC network will continue to skyrocket as home networking appliances are introduced. Also, sharing PC peripherals (such as printers and scanners) between multiple PCs provides cost savings.

- **Evolving in-home applications**: File sizes of typical personal computer applications are growing rapidly with each generation of standard software applications. For example, the file size of a PowerPoint demonstration with the same content has doubled over the last few years. Graphics and digital photography over e-mail are now commonplace. Also, distributing video, MP3 files, and other digital content between information appliances is gaining importance. The end users of a home network demand a different set of requirements than traditional corporate network environments. These requirements include:
    - **Ease of use, low complexity:** Unlike a corporate network, there is no network system administrator in the home. As a result, home networking must be easy to use and simple to install. A home network must be "invisible," providing seamless operation with little or no user intervention or maintenance.
    - **Reliability:** Similar to corporate networks, home networks must be reliable. The network needs to resolve interference from other home networking devices as well as common household appliances, such as microwave ovens and cordless phones.
    - **Scalability**: As consumers begin to buy home networking products, scalability must be considered, even with their first purchase. Scalability results in a lower lifetime cost to the consumer. Buying a network device is not a single product purchase; it establishes the foundation for an entire home network. Consumers who make informed decisions would Avoid purchasing technology that will soon be obsolete. A home network needs to maintain interoperability and accommodate future applications to protect the consumer's initial investment.
    - **Standards compliance:** Industry standards are crucial for enabling mainstream consumer adoption of home networking products. This has proven a powerful factor in fueling the proliferation of corporate networks.
    - **Support for high-bandwidth multimedia content:** Most applications and data types that traverse a corporate network have relatively low bandwidth requirements. In contrast, home networks will have to support all types of bandwidth intensive digital content. Existing and emerging digital devices such as televisions, DVD players, digital video recorders, digital audio/MP3 players, flat-panel displays, digital set-top boxes, and PCs create the need to support multimedia content in the home.

- **Cost:** Consumers, unlike corporations, will never pay high prices for home networks, information appliances, or broadband access. New business models will evolve to grow home networks.

2.3 Home Networking Choices

Broadband home networks can operate over various physical media. These can be organized into three broad groups: structured wiring, existing wiring, and wireless.

2.3.1 No New Wires (Existing Wiring Technologies)

No new wires technologies utilize the existing infrastructure available in most homes across the world. These are based on power lines and phone lines. Advantages of using no new wires are that the consumer does not need to rewire the home and products based on this technology can be cost-effectively deployed immediately.

Existing wiring makes use of electrical, telephone, or coax wiring already installed in the walls. Since structured wiring is quite expensive to install in an existing home, many companies are developing technologies based on the existing wiring in the walls of the home. Existing in-home wirings are all the daisy-chain type and are connected to external service networks.

Power line technologies use the existing electrical wiring. There are many competing technologies [9] (X-10, home plug, Home Phone Line Networking Alliance (homePNA)).

X-10 is an international and open industry standard for communication among devices used for home automation utilizing power line communication (PLC). The uses of X-10 are limited to simple control of lights and power controls of other devices, due to the use of only a single frequency band at a very low transmission rate. Noise is also poorly controlled. It primarily uses power line wiring for signaling and control, where the signals involve brief radio frequency bursts representing digital information. A radio based transport is also defined.

Home plug is one standard that operates via PLC and is capable of achieving near-Ethernet quality performance. This is achieved by splitting the available bandwidth into many channels, each with a different frequency, and optimizing the speed of each channel. Data can be sent simultaneously over the various channels, thus greatly increasing the overall rate of data transfer through the network. Home plug is available for home use in computers, devices, and appliances, and is capable of connecting devices not only to one another, but also to the internet. The original Home Plug 1.0

standards allow for speeds up to 14 Mbps and up to 85 Mbps half-duplex. The current Home Plug A/V standard allows for speeds up to 200 Mbps half-duplex (which may be suitable for applications such as High-definition television (HDTV) and Voice over Internet Protocol (VoIP)).

Phone line uses existing telephone wiring as its medium for communication in home and small office networks. HomePNA has defined a 2.0 specification at about 10 Mbps (the earlier 1.1 specification was about 2 Mb/s). Proprietary technologies operate at much higher speeds (approximately 320Mbps) and are in contention for the 3.1 specification. HomePNA 3 technology enables service providers to meet the growing demand for multimedia services to the home and allows the existing phone and coax wires to be converted into a high-speed multimedia home network for all triple-play IPTV, VoIP and Internet Data by providing data rates up to 320 Mbps. Home users now have high capacity to distribute multiple HDTV and SDTV streams from the network and local sources throughout the home while providing VoIP and Internet access capabilities.

2.3.2 New Wires (Structured Wiring Technologies)

Consumer devices that require high-speed data and video packets are built using Ethernet (IEEE 802.3) which is a longstanding, wired medium for network communications in computing. The system is robust and fast, and it uses existing standards of communication with devices, such as internet protocol (IP). IEEE 1394 (FireWire), unshielded twisted pair (UTP) for telephone and data and coax for video, optical fiber for data, or USB 2.0 (Universal Serial Bus) technologies require additional special wiring around the house, with an additional cost [4] and [9]. Structured wiring provides high bandwidth and excellent security. Fast Ethernet over UTP and gigabit Ethernet over optical fiber are widely used. As HD video moves into homes, it is believed that a “home backbone” network based on structured wiring will be required to interconnect sections of the home.

IEEE1394 which was basically developed as the interface among home appliances such as computer peripherals, video camera, audio device, television, video cassette, and VCR to readily be accessed to the home computer FireWire 400 can transfer data between devices at 100, 200, or 400 Mbps data rates. FireWire 800 allows a transfer rate of 786.432 Mbps with backwards compatibility to the slower rates and 6-pin connectors of FireWire 400.

2.3.3 Wireless Networking

Wireless networking [7] Avoids the use of wires by transmitting through the air over radio channels and is perhaps the most attractive approach for the home, since it Avoids the cost of pulling new wires and the challenges of using existing wiring.

The radio technology used for this purpose can be any of a wide range of systems and standards. Depending on the service area size, a radio technology developed for Wireless Personal Area Networks (WPAN), Wireless Local Area Networks (WLAN), or Wireless Metropolitan Area Networks (WMAN) may be adopted for ad-hoc networks. The coverage radius of a WPAN is roughly in the order of a few meters up to 20 meters. WLAN coverage radius is limited to about 100 meters, while WMAN coverage is in the order of a few kilometers.

For each network type various wireless technologies have been proposed. Some examples are:

- WPAN: Bluetooth (802.15.1), UWB (802.15.3), Zigbee (802.15.4), Infrared Data Association (IrDA)
- WLAN: IEEE 802.11a, IEEE 802.11b, IEEE 802.11g, IEEE 802.11n
- WMAN: IEEE 802.16e

WPAN and WLAN technologies provide connectivity to telecommuters, small office/home office (SOHOs) and hospitals.

2.3.3.1 Wireless Personal Area Networks

WPANs [8] have a very short range of up to 10 meters. They are used to connect mobile devices to send voice and data in order to perform transactions, data transfer, or voice relay functions. They are used in personal computers to replace keyboard and printer cables and connectors. Popular technologies for WPANs are;

802.15.1 Bluetooth [12] is a wireless standard showing significant promise for applications in smart environments, although it is designed primarily for communication between mobile devices, with the goal of unifying the functions of various products from diverse manufacturers. The protocol operates in the license-free ISM band at 2.4 GHz, which is globally considered a free, open frequency band Devices that are Bluetooth enabled will communicate with each other and find out if they need to be on the same network or not. Once two or more devices realize that they need to be on the same network, a piconet is formed. They know this by a predefined address setting either created by the user or input into the device by the manufacturer. This piconet is used to synchronize each participating device so that they use the same hopping sequence. The Bluetooth network, called a piconet, is used to connect up to eight devices. It uses frequency hopping spread spectrum technique implemented with Gaussian frequency shift keying (GFSK). The Bluetooth network is intended for wireless connection between mobile devices, fixed computers, and cellular phones. Bluetooth was designed for short-range personal networking and is being extended for longer range. The main enhancement is the introduction of an enhanced data rate (EDR) of 3.0 Mbps which provides three times faster transmission speed—up to 10 times in certain cases.

Ultrawide Band (802.15.3) (UWB) [10] is based on low-power spread spectrum and also possible technology for use in personal area networks and appears in IEEE 802.15.4a draft PAN standard. The UWB is very similar to how it sounds. A short-range communication technology transmits its signal across a wide frequency range. This wide range is also where other signals would be located. UWB is able to prevent itself from affecting or being affected by these other signals by allowing such a wide band. Narrowband communications are easily detected and avoided when transmitting a signal at low power over a wide frequency. This is the goal of UWB communications. Currently, there are ongoing talks concerning uses for UWB. One can count on very high data rates, upwards of 500 Mbps standard.

Zigbee was created from the 802.15.4 standard [11]. This standard was published to create a low data rate solution that could have long battery life and standard running in the 2.4-GHz ISM band yields 250 kbps with 16 available channels. The current data rate on the 868-MHz Zigbee standard is 20 kbps, with only one static channel available. The goal of ZigBee is to achieve wireless communication of monitoring and control devices using the IEEE 802.15.4 standard. Further, ZigBee seeks to utilize a variety of topologies for including mesh, peer-to-peer, and cluster tree with special attention paid to security. One major advantage of the ZigBee standard is that is highly power-efficient and cost-efficient.

IrDA is not considered a WPAN technology and is not part of the802.15 family [8]. It is listed here because it has a very small distance limitation, which qualifies it as a PAN. Interestingly enough, IR was originally defined in the 802.11 standard. The standard IR has allowed us to connect many devices together quickly and easily for small file transfers. IR is widely deployed on laptops, cell phones, PDAs, and many other hand-held mobile devices. Many people have looked at the security risk of IR; although it has very limited range with almost virtually no security, the risk is low. Unlike the before mentioned standards, IrDA does not make use of radio frequency (RF) waves for communication. Instead, it is based on the infrared (IR) portion of the electromagnetic spectrum. This difference limits IrDA to line-of-sight applications only. Therefore, it is generally only used for short-distance, point-to-point data transfer applications and unlikely that IrDA will become the main standard for wireless communication within smart environments.

Wireless USB based on UWB. Essentially a complement technology for wired USB, wireless USB combines the high throughput of wired USB with the convenience of wireless technology. Designed as a short-range communication, this is not a home networking technology. Wireless USB will deliver speeds up to 480 Mbps at three meters (and up to 110 Mbps at 10 meters) consuming very little power and operates in the 3.1 to 10.6 GHz frequency range and spreads communication over an ultra-wideband of frequencies. This makes it ideal for connecting PC peripherals or wireless USB devices like digital camcorders. Wireless USB is a short-range, high-bandwidth wireless extension to USB that combines the speed and ease-of-use of USB 2.0 with the convenience of wireless technology.

2.3.3.2 Wireless Local Area Networks

WLANs are used to substitute fixed LANs in the range of about 100 meters [6] and [7]. They are used in office buildings and homes to connect devices using a WLAN protocol. Typically, WLANs have a fixed transceiver, which is a base station that connects the WLAN to a fixed network. Popular WLANs include DECT and 802.11 networks.

DECT is a standard for cordless phones that operate in the frequency range from 1880 to 1900 MHz in a range of 50 meters with a net bit rate 32 kbps. It is based on TDMA technology. DECT can also be used for wireless data transfers. Interference-free wireless operation to around 100 meters outdoors. Operates clearly in common congested domestic radio traffic situations. For instance, it is generally immune to interference from Wi-Fi or video senders.

IEEE 802.11 (also known by the brand Wi-Fi) defines the MAC and different physical layers based on radio frequency (RF) and Infrared (IR). Direct Sequence Spread Spectrum (DSSS) and Frequency Hopping Spread Spectrum (FHSS) operating in the 2.4-GHz ISM band are specified for the RF physical layer. The DSSS provides 2 Mbps of peak rate and optional 1 Mbps in extremely noisy environments. On the other hand, the FHSS PHY operates at 1 Mbps with optional 2 Mbps in very clean environments.

The IR PHY supports both 1 Mbps and 2 Mbps for receiving, and 1 Mbps with an optional 2 Mbps bit rate for transmitting. It has become very popular for coordinating wireless networks in home and office environments and is developed for WLANs that cover an office building or a group of adjacent buildings and a family of evolving standards, originally designed for enterprise networking and now moving into the home. These are;

802.11a operates in 5 GHz band, with a maximum raw data rate of 54 Mbps, which yields realistic net achievable throughput in the mid-20 Mbps. It is not interoperable with 802.11b, except if using equipment that implements both standards. Since the 2.4 GHz band is heavily used, using the 5 GHz band gives 802.11a the advantage of less interference. It has approximately 30 meters indoor range. The physical layer technology Orthogonal Frequency Division Multiplexing (OFDM) is used to transfer the data into radio waves. The next technology, which controls the flow and management of the data, is the Carrier Sense Multiple Access with Collision Avoidance (or CSMA/CA). With 802.11a, one has the ability to place multiple access points in the same area, overlapping their coverage cells to provide higher bandwidth. This allows one to have more users per area, sharing a larger amount of bandwidth.

The 802.11b operates at 2.4 GHz has a maximum raw data rate of 11 Mbps. Due to the CSMA/CA protocol overhead, in practice the maximum 802.11b throughput that an application can achieve is about 5.9 Mbps using TCP and 7.1 Mbps using UDP. The dramatic increase in throughput of 802.11b (compared to the original standard) along with substantial price reductions led to the rapid acceptance of 802.11b as the definitive WLAN technology. Typical indoor range is 30 m (100 ft) at 11 Mbps and 90 m (300 ft) at 1 Mbps. Defines a physical layer that provides data rates up to 11 Mbps in the 2.4-GHz ISM radio frequency band. IEEE 802.11b is widely deployed WLAN today. Standard 802.11b (a revision of an original 802.11 standard) subdivides its frequency band of 2.4 to 2.483 GHz into several channels. It supports DSSS technique.

802.11g works in the 2.4 GHz band (like 802.11b) but operates at a maximum raw data rate of 54 Mbps, or about 24.7 Mbps net throughput (like 802.11a) with 30 meters indoor range. 802.11g hardware is compatible with 802.11b hardware. In older networks, however, the presence of an 802.11b participant significantly reduces the speed of an 802.11g network. The modulation scheme used in 802.11g is orthogonal frequency-division multiplexing (OFDM) for the data rates of 6, 9, 12, 18, 24, 36, 48, and 54 Mbps. Even though 802.11g operates in the same frequency band as 802.11b, it can achieve higher data rates because of its similarities to 802.11a. The maximum range of 802.11g devices is slightly greater than that of 802.11b devices, but the range in which a client can achieve the full 54 Mbps data rate is much shorter than that of which an 802.11b client can reach 11 Mbps. Networks using the 802.11g specification employ Complementary Code Keying (CCK) when operating at 802.11b speeds.

802.11n IEEE announced that it had formed a new 802.11 Task Group to develop a new amendment to the 802.11 standard for wireless local-area networks. The real data throughput is estimated to reach a theoretical 540 Mbps (which may require an even higher raw data rate at the physical layer), and should be up to 50 times faster than 802.11b, and up to 10 times faster than 802.11a or 802.11g and it has 50 meters indoor range. 802.11n will work by utilizing multiple wireless antennas in tandem to transmit and receive data. The associated term "MIMO" (Multiple Input, Multiple Output) refers to the ability of 802.11n (and other similar technologies) to coordinate multiple simultaneous radio signals. MIMO increases both the range and throughput of a wireless network. An additional technique employed by 802.11n involves increasing the channel bandwidth.

The HiperLAN [8] Standard in Europe, in the mid 1990s, the HiperLAN standard was developed to provide similar capability as the IEEE 802.11 standards. HiperLAN is primarily used in European countries. There are two specifications: HiperLAN/1 and HiperLAN/2. Both have been adopted by the European Telecommunications Standards Institute (ETSI). HiperLAN/1 provides communication at up to 20 Mbps in the 5 GHz range of the RF spectrum. HiperLAN/2 operates at up to 54 Mbps in the same RF band. HiperLAN/2 is compatible with third-generation WLAN systems for sending and receiving data, images, and voice communication. HiperLAN/2 has the potential and is intended for implementation worldwide in conjunction with similar systems in the 5 GHz RF band. As of 2006 HIPERLAN has largely failed in the marketplace compared to 802.11.

Table 2.2 serves only for rough quality and performance comparison between technologies. The maximum supported bit rate, frequency allocation and typical ranges are important features that determine the appropriateness of each technology for applications to be provided by the wireless network. It is also worth mentioning that ISM frequency bands are license exempt frequency bands.

Table 2.2: Quality and Performance Comparison between Technologies

	Maximum Data rate	**Frequency allocation**	**Modulation**	**Typical range**
Bluetooth	1 Mbps	2.4 GHz	FHSS	10 m
UWB	110 Mbps (at 10m)	3.1-10.6 GHz	THSS OFDM	10-15 m
DECT	32 Kbps	1880-1900 GHz	TDMA	50 m
IEEE 802.11b	11 Mbps	2.4-2.497 GHz	DSSS	50 m (11mbps) 80 m (2 Mbps)
IEEE 802.11g	54 Mbps	2.4-2.497 GHz	DSSS/CCK/ OFDM	50 m (11mbps) 80 m (2 Mbps)
IEEE 802.11a	54 Mbps	5 GHz	OFDM	50 m (11mbps) 60 m (2 Mbps)
IEEE 802.11n (unapproved draft)	540 Mbps	2.4 GHz or 5 GHz	DSSS/CCK/ OFDM	50 m

The emerging picture of the networked home is one of a diverse and ever expanding menagerie of networked devices and device connection technologies. End users, however, are interested in services, not underlying communication protocols. For the networked home to fully realize the benefits of such interconnectivity, devices must be able to discover and communicate with each other to provide compound services across these different device technologies. Device and service discovery methods then become an important component in unlocking the potential of the networked home. Therefore, intelligent management and coordination of discovery methods across these heterogeneous device technologies is paramount to improving the user experience and facilitating networked services deployment in the home.

Networking products that include UPnP™ technology will "just work" when physically connected to a TCP/IP network. UPnP™ technology can work with essentially any networking media technology, wired or wireless. As these devices and PCs are connected with one another, it becomes easier for users to take advantage of innovative new services and applications.

3. UPNP TECHNOLOGY AND DEVICE ARCHITECTURE

3.1 UPnP Technology

UPnP technology is a widely-adopted, standards-based network technology to allow devices from the personal computer and consumer electronics worlds to offer their functionalities and advertise their services to other network devices and has been steadily gaining industry support as the preferred device discovery and control protocol for IP networks in the home and small office.

Devices will work together to provide new usage models that support the various activities in the home, providing new kinds of user interaction paradigms and new levels of automation [16]. The Figure 3.1 shows a sample home network with some of today's UPnP devices.

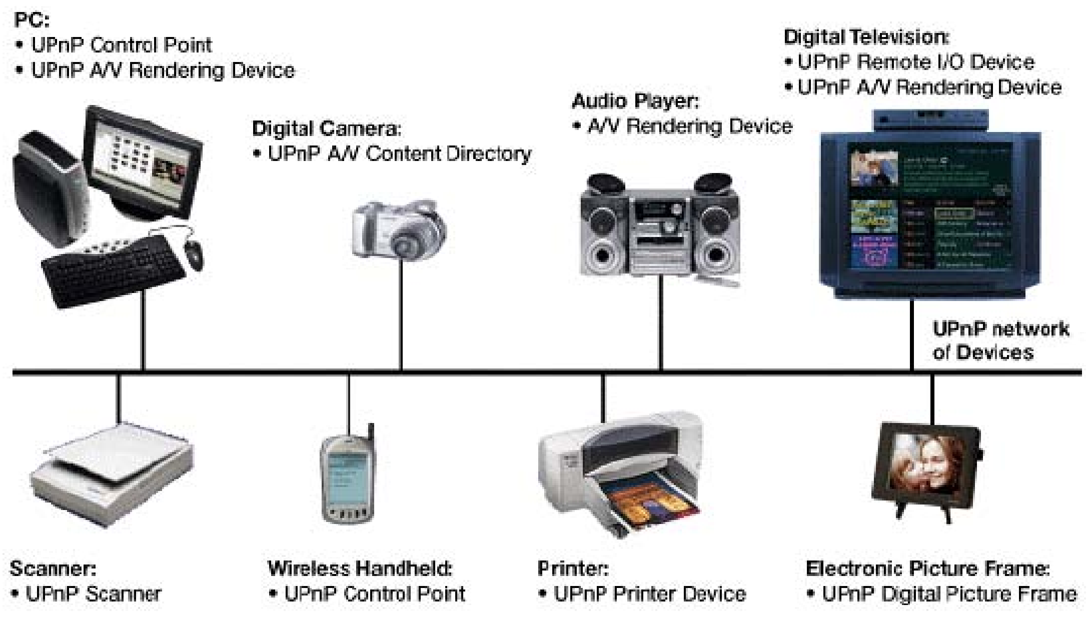


Figure 3.1: UPnP Devices in Digital Home

The scope of UPnP is large enough to encompass many existing, as well as new and exciting scenarios including home automation, printing and imaging, audio/video entertainment, kitchen appliances, automobile networks, and proximity networks in public venues. UPnP technology enables developers to create such products that free their customers from thinking about network configurations, setup, maintenance, software or Internet protocols. UPnP extends this simplicity to include the entire network, enabling discovery and control of devices, including networked devices and services.

UPnP technology, providing maximum user and developer choice and great economics is independent of any particular operating system, programming language, or physical medium (just like the Internet) for device and CP implementation and makes the networking simple and affordable for users for home networks, proximity networks, and networks in small businesses and commercial buildings. It is based on the notion of sending only data, not executable code, between CPs and devices.

UPnP is more than just a simple extension of the Plug and Play peripheral model which makes it a great deal easier to setup, configure, and add peripherals to a PC. It is designed to support zero-configuration, "invisible" networking, and automatic discovery for a breadth of device categories from a wide range of vendors such as network-attached printers, Internet gateways, and consumer electronics equipment. With UPnP, a device can dynamically join a network, obtain an IP address, convey its capabilities, and learn about the presence and capabilities of other devices all automatically. Devices can subsequently communicate with each other directly; thereby further enabling peer to peer networking. DHCP and DNS servers are optional and will be used if available on the network. Furthermore, a device can leave a network smoothly and automatically without leaving any unwanted state behind.

UPnP technology uses no device drivers. Instead, UPnP is built upon existing protocols and technology including TCP, UDP, IP, HTTP, SOAP, and XML and other Internet technologies to enable the devices to automatically connect with one another and work together to make seamless proximity networking, in addition to control and data transfer among networked devices in the home, office, and everywhere in between [17]. The Internet provides highly proven, well understood and open networking technology that is easy to use. IP internetworking is well suited for Universal Plug and Play because it has proven its ability to span different physical media, enable real-world multiple-vendor interoperation and achieve synergy with the Internet and many home and office intranets. By leveraging Internet protocols, UPnP technology is ideally suited for cross-device, cross-network deployment. It can work with any underlying networking technology that supports IP traffic, but it does not require all devices to include an IP stack. Cost, technology or legacy might preclude IP traffic for some media and devices.

By using UPnP protocol bridges [25], devices that communicate with non-IP protocols can participate in UPnP networks.

3.1.1 The UPnP Forum and Technical Foundation

Universal Plug and Play is not only a technology; it is also a cross-industry initiative. The UPnP Forum is the embodiment of that initiative, and its primary mission is to develop device control protocols (DCPs) that describe standard methods for device interaction [13]. The goals of the Forum are to enable the emergence of easily connected devices and to simplify the implementation of networks in the home and corporate environments.

The UPnP Forum is a group of companies and individuals across the industry that intends to play a leading role in the authoring of specifications for UPnP devices and services.

Today, the UPnP Forum consists of more than 822 vendors, including industry leaders in consumer electronics, computing, home automation, home security, appliances, printing, photography, computer networking, and mobile products. UPnP Forum working groups have developed standards for several kinds of network devices, such as printers, cameras, scanners, and Internet gateways. Each of those standards provides an abstraction of what services each kind of device provides to the network. Work in the UPnP Forum continues, with more devices and innovative usage models on the horizon.

The UPnP Implementers Corporation (UIC), a non-profit corporation, has a mission to promote the adoption of UPnP technology by hardware and software manufacturers and administer the UPnP device certification process and the UPnP logo licensing [14]. The UPnP Forum defines and approves the UPnP Device Architecture (UDA) as well as the device and service specifications. The UIC uses these documents in its device certification and logo licensing processes.

UPnP vendors, UPnP Forum Working Committees and the UPnP Device Architecture document define the highest layer protocols used to implement UPnP. Based on the device architecture, the working committees define specifications specific to device types such as VCRs, HVAC systems, and other appliances. Subsequently, UPnP Device Vendors add the data specific to their devices such as the model name, URL, etc.

3.1.2 UPnP Layer

UPnP layer in home networking architecture is shown in Figure 3.2 that UPnP middleware is located in the middle of three-layered home networking architecture. Basically UPnP layer has five components: Zero-Configuration, Discovery & Registration, Device Description, Command & Control, and Events.

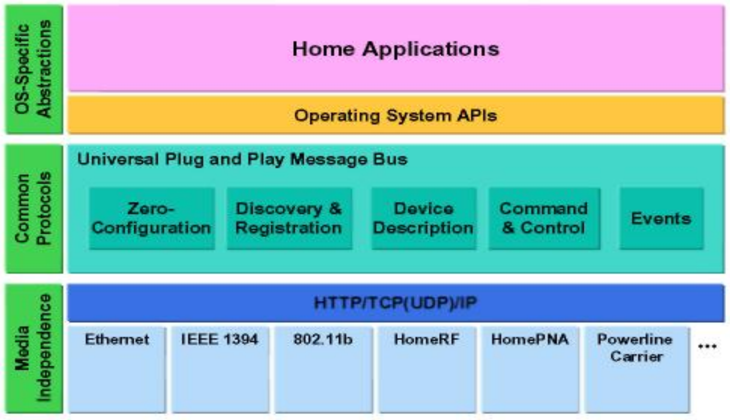


Figure 3.2: UPnP Layer in Home Networking Architecture.

3.1.3 UPnP Protocol Stack

The UPnP Device Architecture defines a schema or template for creating device and service descriptions for any device or service type [20]. Individual working committees subsequently standardize on various device and service types and create a template for each individual device or service type. Finally, a vendor fills in this template with information specific to the device or service, such as the device name, model number, manufacturer name and URL to the service description.

Each UPnP phase has related network protocols. The TCP/IP protocol suite and HTTP provide the basic network connectivity for UPnP. Figure 3.3 shows the basic UPnP protocol stack.

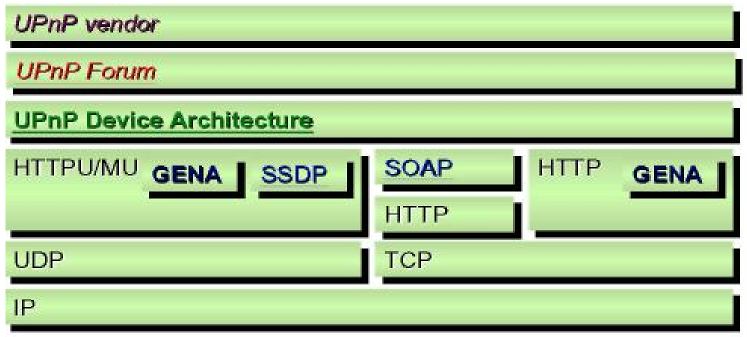


Figure 3.3: The UPnP Protocol Stack

The UPnP protocol stack makes a CP and a device communicate with each other through this protocol. Basically UPnP networking messages are transmitted to each other through HTTP/TCP/IP or HTTP/UDP/IP layers. A device transmit some basic information message on the device capability in XML (Extensible Markup Language) description to multiple CPs on the same network by multicast when it is first connected to a network, or when a CP searches an interesting device on a network by multicasting search messages. The relative protocol is SSDP (Simple Service Discovery Protocol) which is processed through HTTPMU/UDP/IP layers. SSDP protocol is processed through HTTPU/UDP/IP layers when a CP discovers the interesting device and wants to know more about the service capabilities of the device through parsing the description message and getting the URL sent from the device by unicasting the query message or when a CP search the interesting device and the device responds to CP search message. The multicast-oriented protocol stack is shown in Figure 3.4.

The protocol SOAP (Simple Object Access Protocol) invokes the action to control the device via HTTP/TCP/IP layers. There are two types of event notification protocols; the first GENA (General Event Notification Architecture) sends back firmly the result to the action requested by the CP via HTTP/TCP/IP layers. The second GENA notifies the events to the subscribed CPs by multicasting messages via HTTPMU/UDP/IP layers.

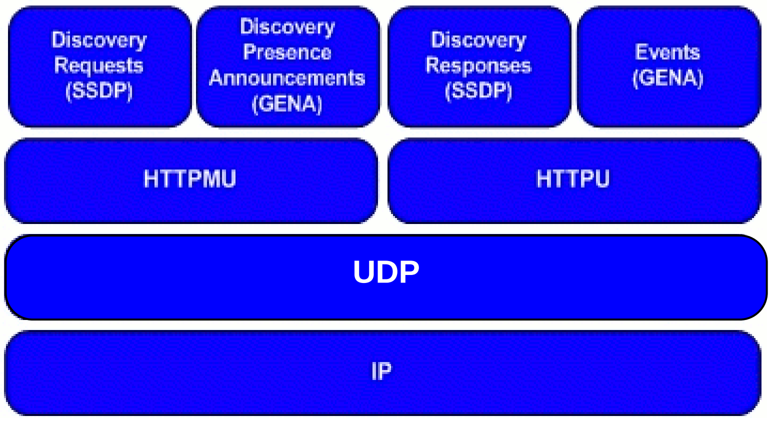

Discovery Requests (SSDP)
Discovery Presence Announcements (GENA)
Discovery Responses (SSDP)
Events (GENA)
HTTPMU
HTTPU
UDP
IP

Figure 3.4: The Multicast-oriented Protocol Stack

3.1.4 UPnP Specific Protocols

3.1.4.1 TCP/IP

The TCP/IP networking protocol stack serves as the base on which the rest of the UPnP protocols are built. By using the standard, prevalent TCP/IP protocol suite, UPnP leverages the protocol’s ability to span different physical media and ensures multiple vendor interoperability. UPnP devices can use many of the protocols in the TCP/IP stack including TCP, UDP, IGMP, ARP and IP as well as TCP/IP services such as DHCP and DNS. Since TCP/IP is one of the most ubiquitous networking protocols, locating or creating an implementation for an UPnP device that is tuned for footprint and/or performance is relatively easy.

3.1.4.2 HTTP, HTTPU, HTTPMU

TCP/IP provides the base protocol stack to provide network connectivity between UPnP devices. HTTP, which is hugely responsible for the success of the Internet, is also a core part of UPnP. All aspects of UPnP build on top of HTTP or its variants. HTTPU (and HTTPMU) are variants of HTTP defined to deliver messages on top of UDP/IP instead of TCP/IP. These protocols are used by SSDP, described next. The basic message formats used by these protocols adheres with that of HTTP and is required both for multicast communication and when message delivery does not require the overhead associated with reliability.

3.1.4.3 SSDP

Simple Service Discovery Protocol (SSDP), as the name implies, defines how network services can be discovered on the network. SSDP is built on HTTPU and HTTPMU and defines methods both for a CP to locate resources of interest on the network, and for devices to announce their availability on the network. By defining the use of both search requests and presence announcements, SSDP eliminates the overhead that would be necessary if only one of these mechanisms is used. As a result, every CP on the network has complete information on network state while keeping network traffic low. Both CPs and devices use SSDP. An UPnP CP, upon booting up, can send an SSDP search request (over HTTPMU), to discover devices and services that are available on the network. The CP can refine the search to find only devices of a particular type (such as a VCR), particular services (such as devices with clock services) or even a particular device. UPnP devices listen to the multicast port. Upon receiving a search request, the device examines the search criteria to determine if they match. If a match is found, a unicast SSDP (over HTTPU) response is sent to the CP. Similarly, a device, upon being plugged into the network, will send out multiple SSDP presence announcements advertising the services it supports. Both presence announcements and unicast device response messages contain a pointer to the location of the device description document, which has information on the set of properties and services supported by the device.

In addition to the discovery capabilities provided, SSDP also provides a way for a device and associated service(s) to gracefully leave the network (bye-bye notification) and includes cache timeouts to purge stale information for self healing.

3.1.4.4 GENA

Generic Event Notification Architecture (GENA) was defined to provide the ability to send and receive notifications using HTTP over TCP/IP and multicast UDP. GENA also defines the concepts of subscribers and publishers of notifications to enable events. GENA formats are used in UPnP to create the presence announcements to be sent using Simple Service Discovery Protocol (SSDP) and to provide the ability to signal changes in service state for UPnP eventing. A CP interested in receiving event notifications will subscribe to an event source by sending a request that includes the service of interest, a location to send the events to and a subscription time for the event notification. The subscription must be renewed periodically to continue to receive notifications, and can also be canceled using GENA.

3.1.4.5 SOAP

Simple Object Access Protocol (SOAP) defines the use of Extensible Markup Language (XML) and HTTP to execute remote procedure calls [19]. It is becoming the standard for RPC based communication over the Internet. By making use of the Internet’s existing infrastructure, it can work effectively with firewalls and proxies. SOAP can also make use of Secure Sockets Layer (SSL) for security and use HTTP’s connection management facilities, thereby making distributed communication over the Internet as easy as accessing web pages. Much like a remote procedure call, UPnP uses SOAP to deliver control messages to devices and return results or errors back to CPs. Each UPnP control request is a SOAP message that contains the action to invoke along with a set of parameters. The response is a soap message as well and contains the status, return value and any return parameters.

3.1.4.6 XML

Extensible Markup Language (XML) is the universal format for structured data on the Web and is a way to place nearly any kind of structured data into text files [18] and [36]. XML looks a lot like HTML in that it uses tags and attributes. Actually, it is quite different in that these tags and attributes are not globally defined as to their meaning, but are interpreted within the context of their use. These features of XML make it a good fit for developing schemas for various document types. The use of XML as a schema language is defined by the W3C. XML is a core part of UPnP used in device and service descriptions, control messages and eventing.

3.2. UPnP Device Architecture

3.2.1 Devices

A UPnP device can be thought of as a logical container with a unique type. Each UPnP device may offer any number of services, each service with its own unique service type. For identification purposes, the device must host an XML device description document that lists specific properties about the device, the services associated with the device, and the nested devices. The device description document must also include a Uniform Resource Locator (URL) for the service description. The service description is an XML document that lists the actions and state variables that apply to a specific service offered by the device. By itself, a device does nothing more than provide self-describing information such as the manufacturer, model name, and serial number. A device's services provide the real functionality. In addition to the set of services, the device description also lists the properties (such as device name and icons) associated with the device.

A UPnP device may also contain other devices. That logical composition of devices gives the designer flexibility when determining how to implement the device. It also allows the embedded device to be discovered and used independently from the main container device. Figure 3.5 shows a detailed Unified Modeling Language (UML) class diagram for a UPnP device.

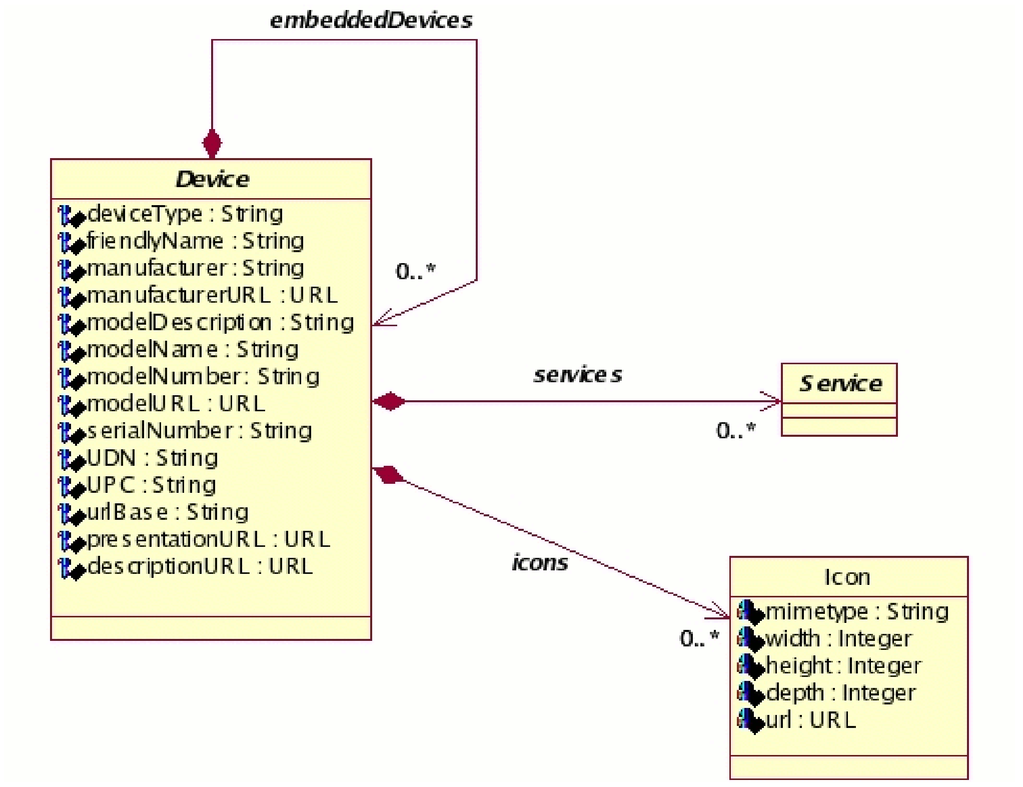


Figure 3.5: Class Diagram: The UPnP Device

3.2.2 Services

Each service in a UPnP device may contain any number of actions, each action having a set of input parameters and an optional return value. Each action also has a name and an optional set of arguments. Each argument has a name, a value, and a direction. The direction may be either input or output (but not both), depending on whether the argument is passed into the action or is returned from the action to the caller. Each action may also have a return value that provides the result of the action.

Those familiar with interface-oriented object model systems, such as Microsoft COM or CORBA, can draw an analogy: an UPnP device is similar to an object, the services are like interfaces supported by the object, and the actions in a service are like functions in an interface.

Each UPnP service also has a service type Uniform Resource Identifier (URI) that uniquely identifies the service. Standard service types are defined by a UPnP Forum working committee. Every service also has a serviceID URI that uniquely identifies the service among all of a device's services. No two services may share the same serviceID.

Every service contained in a device maintains three URLs that provide the information necessary for CPs to communicate with that service:

- The ControlURL is where CPs post requests to control the service. A UPnP device vendor specifies one for each device.
- The EventSubURL is where CPs post requests to subscribe to events. There is one EventSubURL for each service in a device. If a service has no evented variables, that service does not have eventing. If a service does not have eventing, the EventSubURL element must be present but should be empty.
- The DescriptionURL tells CPs the location to retrieve the service description document from. The service description XML document is returned via an HTTP GET request. Sometimes even after recognizing a UPnP Forum device type, a CP discovers individual service description documents. Since some service interfaces are optional, not all vendors may implement a particular capability. For example, a UPnP-enabled printer that does not support color printing will likely choose not to implement actions to adjust color properties.

The services that a device must implement are determined by the device's type. The working committees of the UPnP Forum standardize the set of services a device type must support.

Each service may also maintain variables that represent that service's current status. A service may have zero or more such variables. Each state variable has a name, a type, a default value, and a current value. It also has a set of allowed values to describe the range of permissible variable values. Any state variable can trigger events on state changes; when such events occur is determined by a service's implementer.

If a variable triggers an event to indicate a state, that variable is said to be evented. Every input argument to an action is associated with one of the service's state variables, termed the argument's related state variable. One of an action's arguments may be designated as the action's return value. The return value provides the result of the action to the caller. Figure 3.6 shows a UML class diagram for a UPnP service.

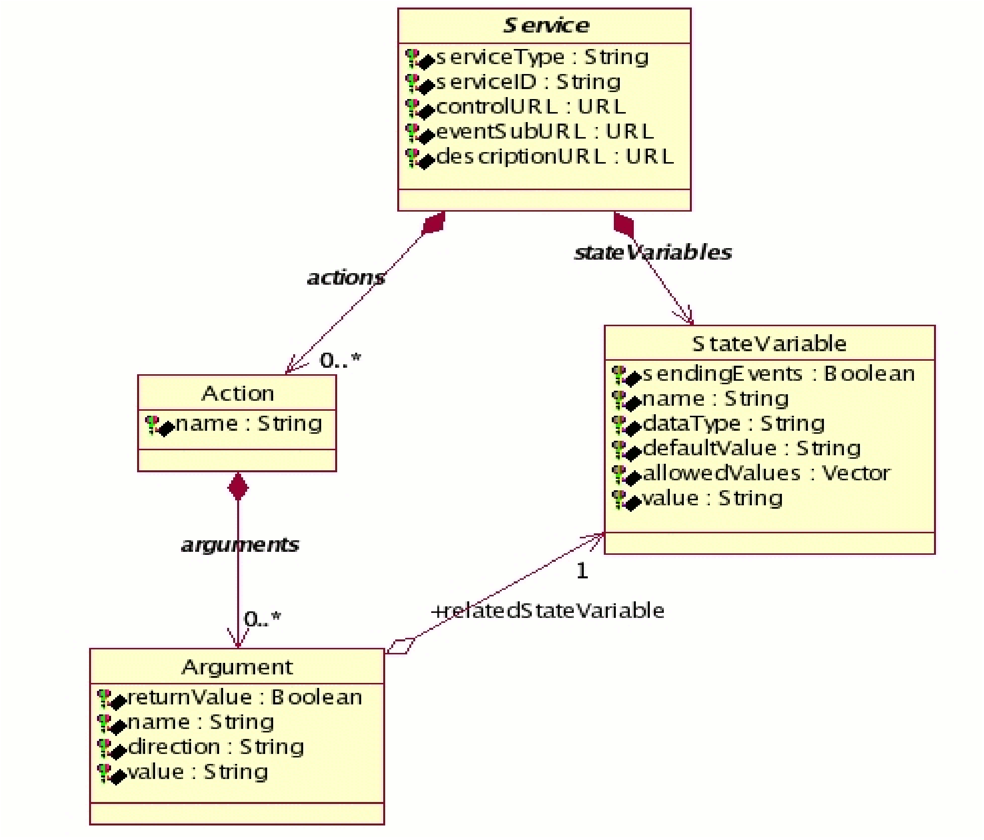


Figure 3.6: Class Diagram: The UPnP Service

UPnP CPs can subscribe to receive indications of state variable changes from a service. When a service detects a change to a state variable, that service notifies any registered CPs of that change. For example, a service that renders audio tracks might keep a state variable of the URL of the current track being played. When the track changes, the service sends a “state change” notification with the new URL to all CPs that have registered interest in receiving events from that service.

3.2.3 Control Points

An UPnP CP is a controller (which may be embedded in a UPnP device but typically a PC) capable of discovering and controlling other devices. In client/server computing terms, the CP is the client and the device is the server. CPs invoke actions on services, providing any required input parameters and receiving any output parameters and possibly a return value. Figure 3.7 shows a CP invoking an action on a UPnP-enabled device that implements a single UPnP device type containing two services.

A CP is able to retrieve descriptions of devices and the services they offer, invoke the advertised actions to use the service and subscribe to a service to receive event messages.

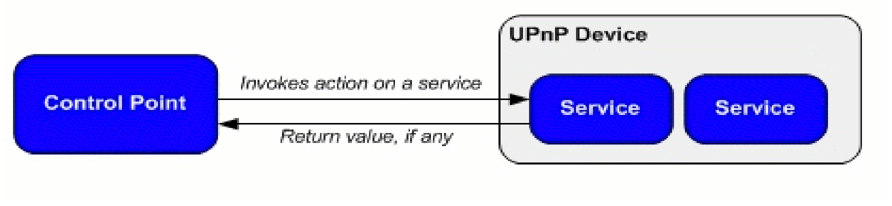


Figure 3.7: A CP Invoking an Action on a Service

As the figure illustrates, a CP discovers devices, invokes actions on a device's services, and subscribes to event notifications. A device, on the other hand, responds to invoked actions, sends events when state variables change, and supports a Web page for administrative control (a presentation page in UPnP terminology).

Figure 3.8 shows the interaction of devices and CPs.

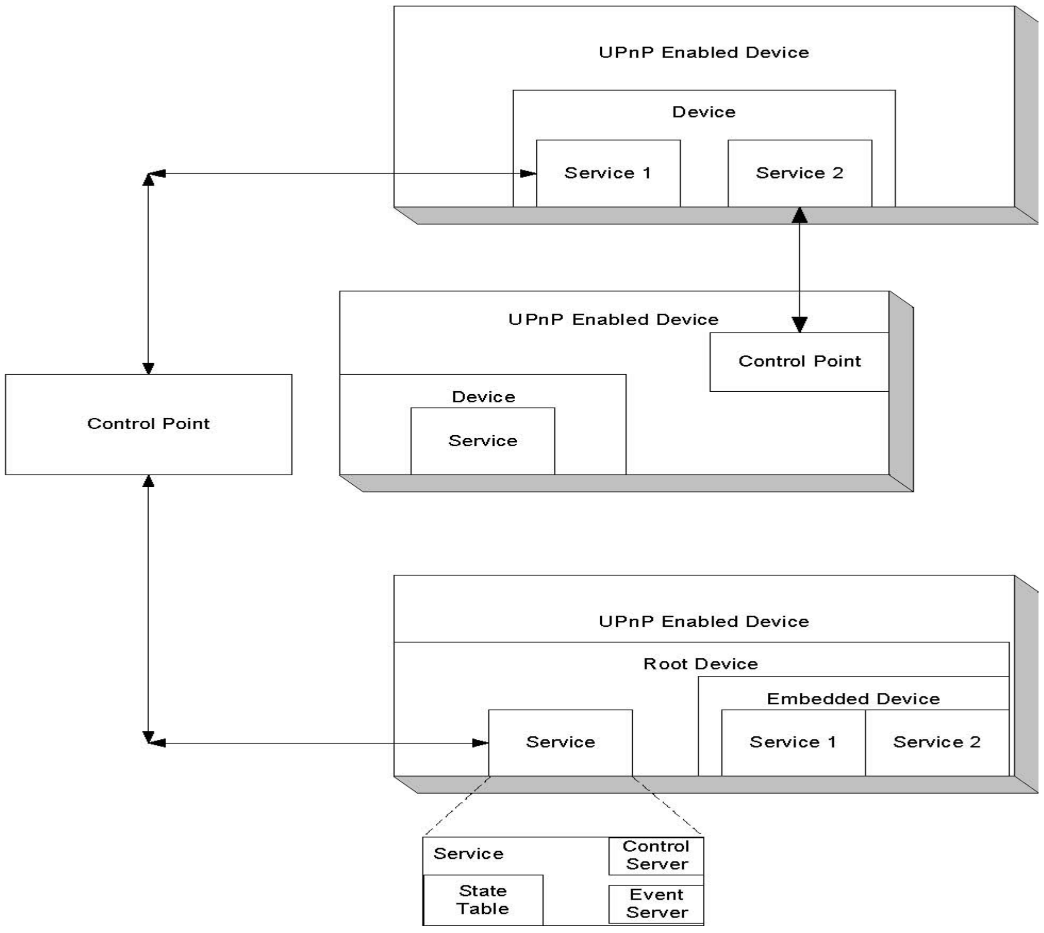


Figure 3.8: UPnP CPs, Devices, and Services

Figure 3.9 diagram shows the sequence of interactions between a CP and device during the UPnP phases. A description of each step follows the diagram. Due to the asynchronous nature of the sequence, the interactions do not necessarily happen in the order shown. The control and eventing steps can happen in any order.

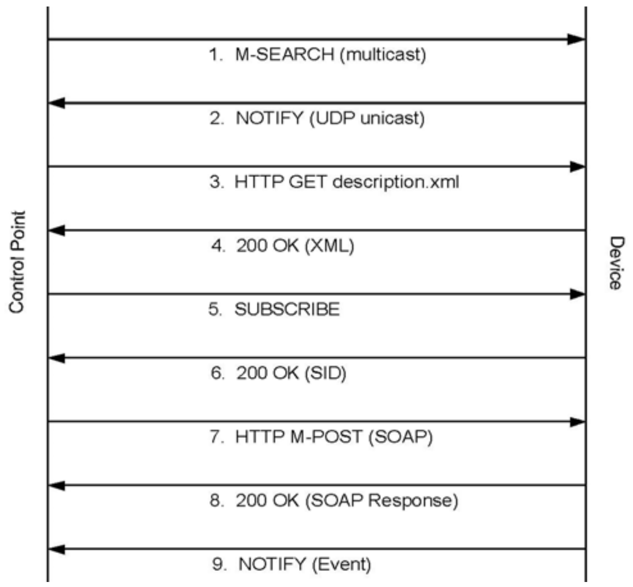


Figure 3.9: The Sequence of Interactions between a CP and Device

3.2.4 The UPnP Phases

UPnP is built upon existing protocols and formats including HTTP, XML, SOAP, the Document Object Model, and IP multicast. Those elements are used together to define a framework to enable the definition, discovery, and control of network devices. The UPnP networking procedure can be divided into six steps from step0 to step5 as shown in Figure 3.10.

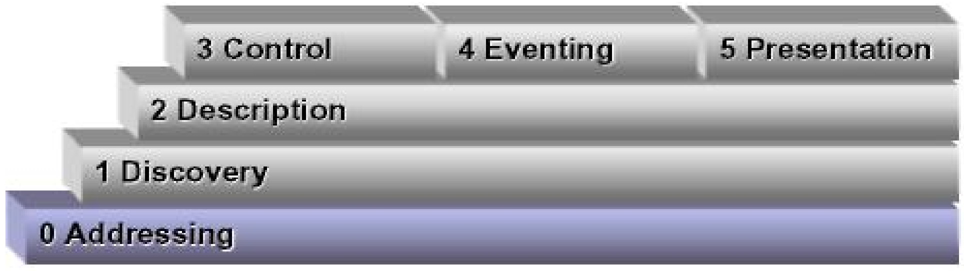


Figure 3.10: UPnP Networking Procedure

The UPnP Device Architecture is a framework that defines the protocols for communication between controllers, or CPs, and devices. UPnP functionality involves five processes:

- Addressing—the device joins the network, acquiring a unique address that others can use to communicate with it. The addressing protocol built in to UPnP devices allows them to join an IP network dynamically and to acquire an address without user configuration. In unmanaged or ad-hoc network, network nodes themselves make up the network. In managed network, network allows devices to acquire an IP address from a DHCP server on the network.
- Discovery—When a UPnP device is added to the network, the discovery protocol allows the device to advertise its presence to CPs by using SSDP. The information exchanged between the device and the CP is limited to discovery messages that provide basic information about the devices and their services (e.g., their types, identifiers, and pointers to more detailed information).
- Description—Using the URL provided in the discovery process, a CP receives XML information about the device, such as manufacturer information like make, model, serial number, and URLs to vendor specific Web sites. In addition, the description process can also include a list of embedded devices, embedded services, and URLs used to access device features.
- Control—given knowledge of a device and its services, CPs use URLs provided during the description process to access additional XML information that describes actions to which the UPnP device services respond, along with parameters for each action. Control messages are formatted in XML and use SOAP.
- Eventing—when a CP subscribes to a service, the service publishes updates to the CP to announce changes in device status when one or more of the state variables that are evented change. Event messages are formatted in XML and use GENA protocol.
- Presentation—if an UPnP device has an URL for presentation, then the CP can retrieve a page from this URL, load the page into a browser and, depending on the capabilities of the page, allow a user to access interface control features, device, or service information, or any device-specific abilities implemented by the manufacturer.

3.2.4.1 Addressing

Addressing is the process by which a device automatically acquires an IP address. It is the first step in the operation of a UPnP device. Addressing allows a device to join the network and to prepare to communicate with other UPnP devices and CPs. The addressing protocols built in to UPnP devices allow devices to join an IP network dynamically and to acquire an address without user configuration.

Addressing takes into account whether a device is operating in an unmanaged or managed network. An unmanaged, or ad-hoc, network is a network where there are no pre-existing infrastructure devices (or they are currently inoperable), and the network nodes themselves make up the network. A managed, or infrastructure, network allows devices to acquire an IP address from a Dynamic Host Configuration Protocol (DHCP) server on the network. As a result, UPnP devices must support both the DHCP and the dynamic configuration of private link-local addresses (Auto-IP).

A device first attempts to contact a DHCP service to acquire an address. If it fails to locate a DHCP server, the device uses Auto-IP, which enables devices to select addresses without a DHCP server being present to assign that address. It randomly chooses an address in the 169.254/16 range, and tests it using an Address Resolution Protocol (ARP) probe to determine if it is already in use. If so, another address is chosen and tested until an unused address is found. It then periodically checks for the existence of a DHCP server; if a server responds, the device uses the assigned address, and stops using the address selected by Auto-IP after a period of parallel use to complete interactions in progress. The addresses are taken from a set of well-known, non-routable addresses (addresses in range will not cross gateways). DHCP is a UDP-based protocol, while Auto-IP uses the Address Resolution Protocol (ARP). In addition to numeric IP addresses, names can be used to access devices if they are identified in a Domain Name Service (DNS) server. Device names could be registered manually, dynamically or by the DHCP server.

3.2.4.2 Discovery

Discovery is the first step in UPnP networking. When a device is added to the network, the UPnP discovery protocol which is SSDP allows that device to advertise its services to CPs on the network. Similarly, when a CP is added to the network, the UPnP discovery protocol allows that CP to search for devices of interest on the network. The fundamental exchange in both cases is a discovery message containing a few, essential specifics about the device or one of its services, e.g., its type, universally unique identifier, and a pointer to more detailed information. Figure 3.11 shows the Discovery step in UPnP networking.

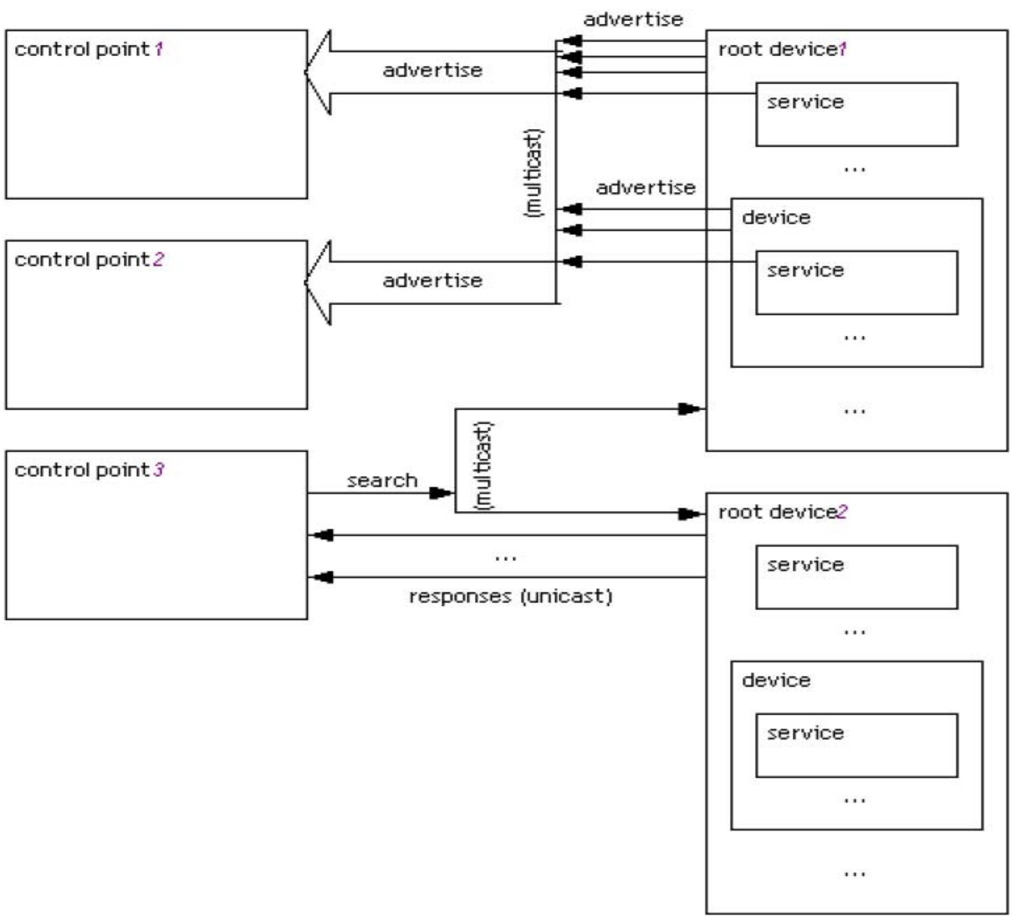


Figure 3.11: Discovery Step in UPnP Networking

Discovery: Advertisement

To send (and receive) advertisements, devices (and CPs) use Figure 3.12.

UPnP vendor
UPnP Forum
UPnP Device Architecture
SSDP
HTTPMU
UDP
IP

Figure 3.12: Subset of Overall UPnP Protocol Stack

Devices don’t have to wait for a CP to search for their services. When a device is added to the network, that device periodically advertises itself and its services on the network. They can advertise their device availability by means of the SSDP NOTIFY command on the 239.255.255.250:1900 multicast address. When a device is added to the network, it must send a multicast request with method NOTIFY and “ssdp: alive” in the NTS header in the following format. Values in italics are placeholders for actual values.

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age = seconds until advertisement expires
LOCATION: URL for UPnP description for root device
NT: search target
NTS: ssdp:alive
SERVER: OS/version UPnP/1.0 product/version
USN: advertisement UUID
```

When CPs see this NOTIFY multicast, they can request the device’s description document using a standard HTTP GET request to the URL in the NOTIFY message.

A device includes a URL of its device description XML document in its advertisement and discovery responses. That URL provides CPs with the information those CPs need to retrieve the device and service descriptions. The description documents are retrieved by CPs and parsed to understand all about that device. A device may consist of other devices, each with its own services. If a root device has d embedded devices and s embedded services but only k distinct service types, this works out to 3+2d+k requests. If a particular device or embedded device contains multiple instances of a particular service type, it is only necessary to advertise the service type once (rather than once for each instance). This advertises the full extent of the device's capabilities to interested CPs. These messages must be sent out as a series with roughly comparable expiration times; order is unimportant, but refreshing or canceling individual messages is prohibited. Devices are identified both by type and by a unique identifier. Services are identified by their type.

Advertisement messages indicate the lifetime of the advertisement. If a device remains available, it retransmits its advertisements well before the life time expiration. If a device or service becomes unavailable and an orderly shutdown is possible, previous advertisements are cancelled by sending cancellation messages. Each multicast request must have method NOTIFY and “ssdp:byebye” in the NTS header in the following format. Values in italics are placeholders for actual values.

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
NT: search target
NTS: ssdp:byebye
USN: uuid:advertisement UUID
```

Otherwise, advertisements eventually expire by themselves. The lifetime for advertisements is selected to balance the need to minimize network traffic with the desire to maximize the currency of device status.

Discovery: Search

To search for devices (and be discovered by CPs), CPs (and devices) use the subset of the overall UPnP protocol stack in Figure 3.13.

UPnP vendor	
UPnP Forum	
UPnP Device Architecture	
SSDP	
HTTPU (unicast)	HTTPMU (multicast)
UDP	
IP	

Figure 3.13: UPnP Discovery Search Protocol Stack

To search for devices or services on the network, CPs use the HTTP M-SEARCH command in the following format multicast to the address 239.255.255.250:1900 over UDP.

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: seconds to delay response
ST: search target
```

Any device on the network that matches the criteria the CP is searching for, issues a unicast UDP reply that includes the URL to its description document. If a CP receives one or more acceptable replies, it moves into the description phase. Figure 3.14 illustrates the steps taken by an UPnP CP application to discover a device's capabilities.

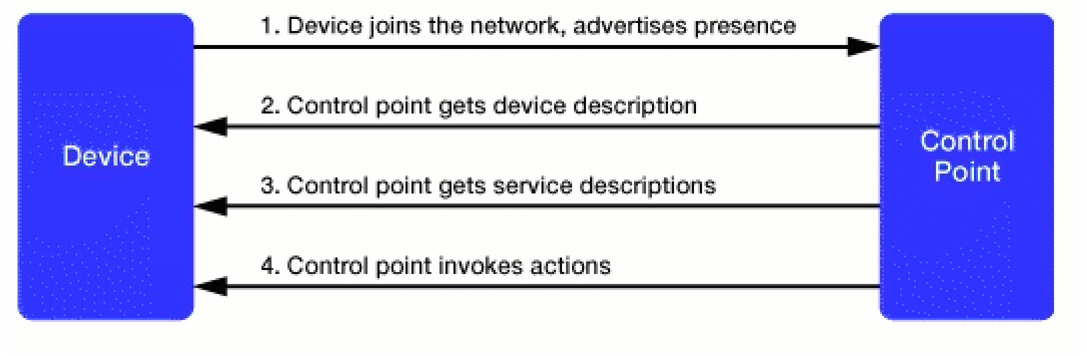

1. Device joins the network, advertises presence
2. Control point gets device description
3. Control point gets service descriptions
4. Control point invokes actions
Device
Control Point

Figure 3.14: Retrieving Device and Service Description

When a CP sends out a search request, it includes the amount of time it is willing to wait in an SSDP header. Matching devices will wait a random time between zero and the number of seconds the CP indicated before responding. If the CP does not receive any replies when its search time has expired, it can assume that there are currently no matching devices on the network. To be found, a device must send a UDP response to the source IP address and port that sent the request to the multicast channel. Devices respond if the ST header of the M-SEARCH request is “ssdp:all”, “upnp:rootdevice”, “uuid:” followed by a UUID that exactly matches one advertised by the device. Responses to M-SEARCH are intentionally parallel to advertisements, and as such, follow the same pattern as listed for NOTIFY with ssdp:alive (above) except that the NT header there is an ST header here. The response must be sent in the following format. Values in italics are placeholders for actual values.

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = seconds until advertisement expires
DATE: when response was generated
EXT:
LOCATION: URL for UPnP description for root device
SERVER: OS/version UPnP/1.0 product/version
ST: search target
USN: advertisement UUID
```

3.2.4.3 Description

When a CP locates a service it wants to know more about, it requests the description document. To learn more about the device and its capabilities, the CP must retrieve the UPnP description for the device using the URL provided by the device in the discovery message. Then, the CP must retrieve one or more service descriptions using the URL(s) provided in the device description. This is a simple HTTP-based process and uses the subset of the overall UPnP protocol stack in Figure 3.15.

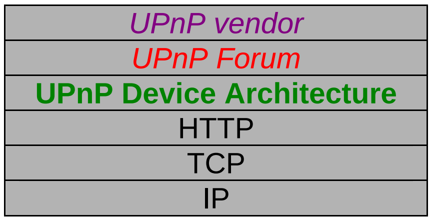


Figure 3.15: UPnP Description Protocol Stack

The description includes a device description describing the physical and logical containers, and one or more service descriptions describing the capabilities exposed by the device. Retrieving an UPnP device or service description is simple: the CP issues an HTTP GET request on the URL in the discovery message.

```
GET path to description HTTP/1.1
HOST: host for description:port for description
ACCEPT-LANGUAGE: language preferred by CP
```

After a CP sends a request, the device takes the second step and responds with a copy of its description. Including expected transmission time, a device must respond within 30 seconds. If it fails to respond within this time, the CP should re-send the request. A device must send a response in the following format.

```
HTTP/1.1 200 OK
CONTENT-LANGUAGE: language used in description
CONTENT-LENGTH: Bytes in body
CONTENT-TYPE: text/xml DATE: when responded
```

On the server side, the device runs a standard HTTP server—either a full Web server such as Apache or a mini-server. Many of the elements in the description document are URLs. Retrieving an UPnP service description is a similar process that uses a URL within the device description

Description allows devices to list the functionality they provide. Descriptions of devices and their services are contained in XML-based description documents. The device description document contains device information such as manufacturer, make, model, and serial number; a list of services provided by the device; and a list of embedded devices. The following device (Figure 3.16) and service description (Figure 3.17) documents have a listing with placeholders (in italics) for actual elements and values. Some of these placeholders would be specified by a UPnP Forum working committee (colored red) or by a UPnP vendor (purple). For a non-standard device, all of these placeholders would be specified by an UPnP vendor.

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    <deviceType>urn:schemas-upnp
     org:device:deviceType:v</deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
    <manufacturerURL>URL to manufacturer
     site</manufacturerURL>
    <modelDescription>long user-friendly
     title</modelDescription>
    <modelName>model name</modelName>
    <modelNumber>model number</modelNumber>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
    <iconList>
      <icon>
        <mimetype>image/format</mimetype>
        <width>horizontal pixels</width>
        <height>vertical pixels</height>
        <depth>color depth</depth>
        <url>URL to icon</url>
      </icon>
      XML to declare other icons, if any, go here
    </iconList>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-
         org:service:serviceType:v</serviceType>
          <serviceId>urn:upnp-
org:serviceId:serviceID</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      Declarations for other services defined by a UPnP Forum
       working committee (if any)go here
      Declarations for other services added by UPnP vendor (if
      any) go here
    </serviceList>
    <deviceList>
    Description of embedded devices defined by a UPnP Forum
     working committee (if any)go here
    Description of embedded devices added by UPnP vendor (if
     any) go here
    </deviceList>
    <presentationURL>URL for presentation</presentationURL>
  </device>
</root>
```

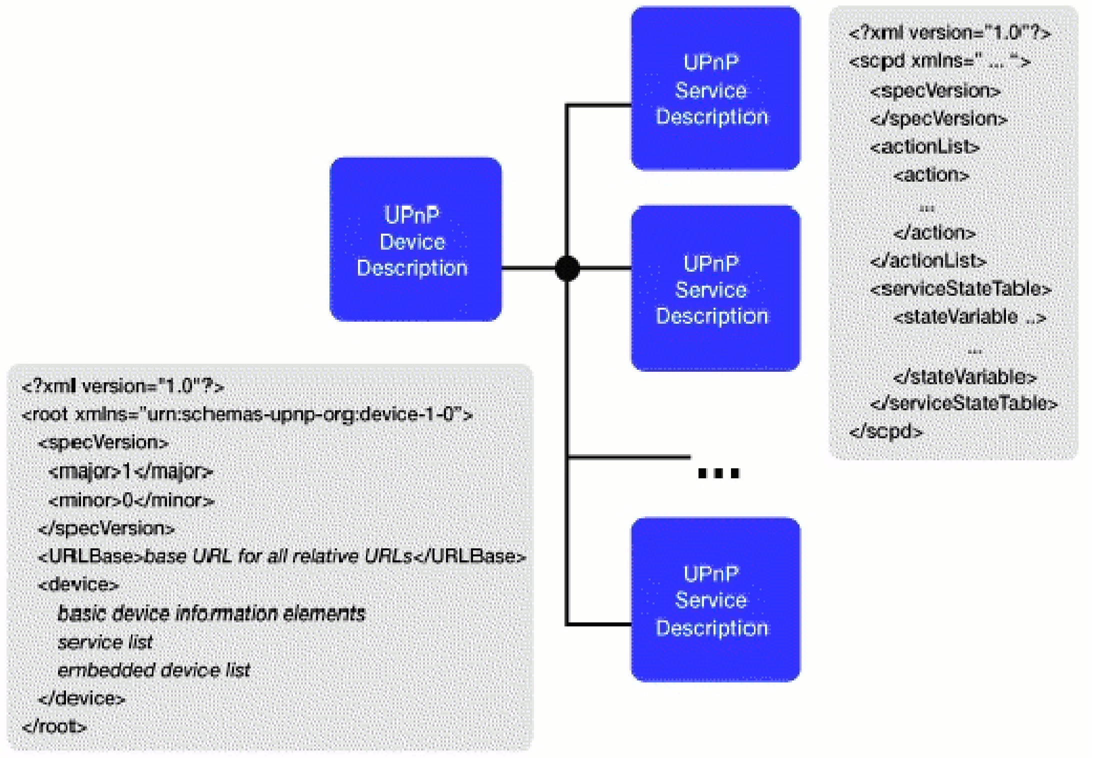
Figure 3.16: Device Description Document

A service description document contains detailed information about a device's service, the actions that service provides, and the service's parameters and return value.

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
 <specVersion>
   <major>1</major>
   <minor>0</minor>
 </specVersion>
 <actionList>
  <action>
    <name>actionName</name>
    <argumentList>
      <argument>
        <name>formalParameterName</name>
        <direction>in xor out</direction>
        <retval />
        <relatedStateVariable>stateVariableName
         </relatedStateVariable>
      </argument>
        Declarations for other arguments defined by UPnP Forum
       working committee (if any)go here
    </argumentList>
  </action>
 Declarations for other actions defined by UPnP Forum working
  committee (if any) go here
 Declarations for other actions added by UPnP vendor (if any)
  go here
 </actionList>
 <serviceStateTable>
   <stateVariable sendEvents="yes">
   <name>variableName</name>
   <dataType>variable data type</dataType>
   <defaultValue>default value</defaultValue>
   <allowedValueList>
     <allowedValue>enumerated value</allowedValue>
     Other allowed values defined by UPnP Forum working
      committee (if any) go here
   </allowedValueList>
   </stateVariable>
   <stateVariable sendEvents="yes">
     <name>variableName</name>
     <dataType>variable data type</dataType>
     <defaultValue>default value</defaultValue>
     <allowedValueRange>
       <minimum>minimum value</minimum>
       <maximum>maximum value</maximum>
       <step>increment value</step>
     </allowedValueRange>
   </stateVariable>
   Declarations for other state variables defined by UPnP
    Forum working committee (if any) go here
   Declarations for other state variables added by UPnP vendor
    (if any) go here
 </serviceStateTable>
</scpd>
```

Figure 3.17: Service Description Document

The description documents are used by CPs in the device discovery process to learn more about a device. Figure 3.18 illustrates the device and service description hierarchy.



```
<?xml version="1.0"?>
<scpd xmlns=" ... ">
  <specVersion>
  </specVersion>
  <actionList>
    <action>
    ...
    </action>
  </actionList>
  <serviceStateTable>
    <stateVariable ..>
    ...
    </stateVariable>
  </serviceStateTable>
</scpd>
```

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    basic device information elements
    service list
    embedded device list
  </device>
</root>
```

Figure 3.18: The UPnP Device and Service Description Hierarchy

UPnP descriptions use XML syntax and are based on a standard UPnP device template or service template. Device and service templates are produced by the UPnP Forum and are derived from the UPnP template language. The template language itself is written in XML syntax and is derived from an XML schema language. Because the template language, templates, and descriptions are all machine-readable, automated tools can be used to ensure that the latter two have all required elements, are correctly nested and have values of the correct data types.

UPnP vendors can differentiate their devices by extending services, embedding other devices and including additional UPnP services, actions, or state. When a CP retrieves a device's description, these added features are exposed to the CP for control and eventing.

3.2.4.4 Control

Once a CP discovers a device and retrieves its description document, it may want to control one or more of the services contained in the device. To invoke actions and poll for values, CPs (and devices) use the following subset of the overall UPnP protocol stack in Figure 3.19.

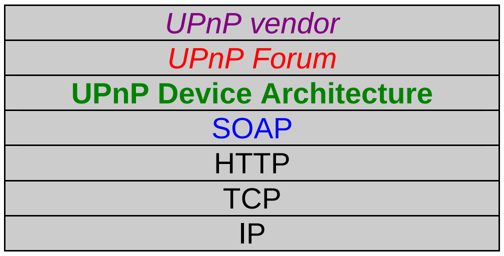


Figure 3.19: UPnP Control Phase Protocol Stack

The Simple Object Access Protocol (SOAP) allows a CP to query or change elements in a service's state table. SOAP uses XML for remote procedure call to specify what actions to take. To invoke an action on a device's service, a CP must send an action request with method POST transported over TCP in the following format. Values in italics are placeholders for actual values.

```
POST path of control URL HTTP/1.1
HOST: host of control URL:port of control URL
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-upnp-
org:service:serviceType:v#actionName"
```

```
<?xml version="1.0"?>
<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <s:Body>
       <u:actionName xmlns:u="urn:schemas-upnp
       org:service:serviceType:v">
          <argumentName>in arg value</argumentName>
          other in args and their values go here, if any
       </u:actionName>
      </s:Body>
</s:Envelope>
```

The CP creates the XML document and posts it to the control URL for the service, as specified in the description document. The CP can request current values and make changes to the service's state table. On the server side, the control server waits for control requests. The control server is an HTTP -like server implementing the SOAP protocol. A device can operate more than one control server depending on the combination of services provided by the device.

The SOAP message is the basic unit of communication between peers. SOAP messages are written in XML, making SOAP platform independent: any system capable of creating and parsing XML documents can send and receive SOAP messages. The power of XML allows SOAP messages to be fairly complex in structure and to transmit highly complex data types. SOAP consists of four parts:

- **The SOAP envelope:** An XML schema that defines a framework for describing what is in a message, how to process it, and whether that processing is optional or mandatory.
- **The SOAP encoding rules:** Another XML schema that defines a set of rules for expressing instances of application-defined data types.
- **The SOAP binding:** A convention for using different transport protocols. SOAP can potentially be used in combination with a variety of other transport protocols (however SOAP messages are most commonly carried by HTTP).
- **The SOAP RPC representation:** A convention for representing remote procedure calls and responses.

SOAP uses the HTTP Extension framework to extend HTTP. To ensure that the introduced SOAP Action is not confused with other HTTP extensions, SOAP conforms to the HTTP Extension framework by specifying a unique URI in the MAN header and attaching the prefix M- to the POST method. Using the M-POST method requires the HTTP server to find and understand the URI in the MAN header and to understand the SOAP Action header.

The SOAP specification requires that requests must be first attempted without the MAN header or M- prefix. If the requests fail with “405 Method Not Allowed”, then a CP must send a second request with method M-POST and MAN in the following format. If that request fails with “501 not implemented” or “510 not extended”, then the request fails. Values in italics are placeholders for actual values.

```
M-POST path of control URL HTTP/1.1
HOST: host of control URL:port of control URL
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
MAN: "http://schemas.xmlsoap.org/soap/envelope/"; ns=01
01-SOAPACTION: "urn:schemas-upnp-org:service:serviceType:v#actionName"
```

(Message body for request with method M-POST is the same as body for request with method POST)

A service has 30 seconds to complete the action and respond to the CP, including expected transmission time. According to the UPnP device architecture, actions that are expected to take longer than this should return early and send an event when the action completes. A service should use the following format:

```
HTTP/1.1 200 OK
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
DATE: when response was generated
EXT:
SERVER: OS/version UPnP/1.0 product/version
<?xml version="1.0"?>
<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
    <s:Body>
    <u:actionNameResponse xmlns:u="urn:schemas-upnp-
    org:service:serviceType:v">
       <argumentName>out arg value</argumentName>
        other out args and their values go here, if any
    </u:actionNameResponse>
    </s:Body>
</s:Envelope>
```

In addition to invoking an action on a device’s service, CPs may also directly query the service for the value of the state variable. To do this, a CP can use “QueryStateVarible” action. All services support this action implicitly. A CP can use “QueryStateVarible” to get the value of a single variable. The QueryStateVariable action has been deprecated by the UPnP Forum and should not be used by CPs except in limited testing scenarios due to inconsistent results.

3.2.4.5 Eventing

A UPnP service description includes a list of actions that the service responds to and a list of variables that model the state of the service at run time. The service publishes updates, in the form of event messages, when these variables change and a CP may subscribe to receive this information. To send and receive subscription and event messages, CPs and services use the following subset of the overall UPnP protocol stack in Figure 3.20.

UPnP vendor
UPnP Forum
UPnP Device Architecture
GENA
HTTP
TCP
IP

Figure 3.20: UPnP Eventing Protocol Stack

UPnP employs an event notification system using a publisher/subscriber model as shown in Figure 3.21 in which CPs may subscribe to events sent by a service, and services publish event notifications to subscribers. An event is a message from a service to subscribed CPs. Events keep CPs informed of changes in state associated with a service. CPs subscribe to events, and the service notifies interested CPs of events. A CP interested in receiving notification of state variable changes subscribes to an event source by sending a subscription request that includes:

- The service of interest
- A URL to which the events can be sent
- A subscription time for the event notification

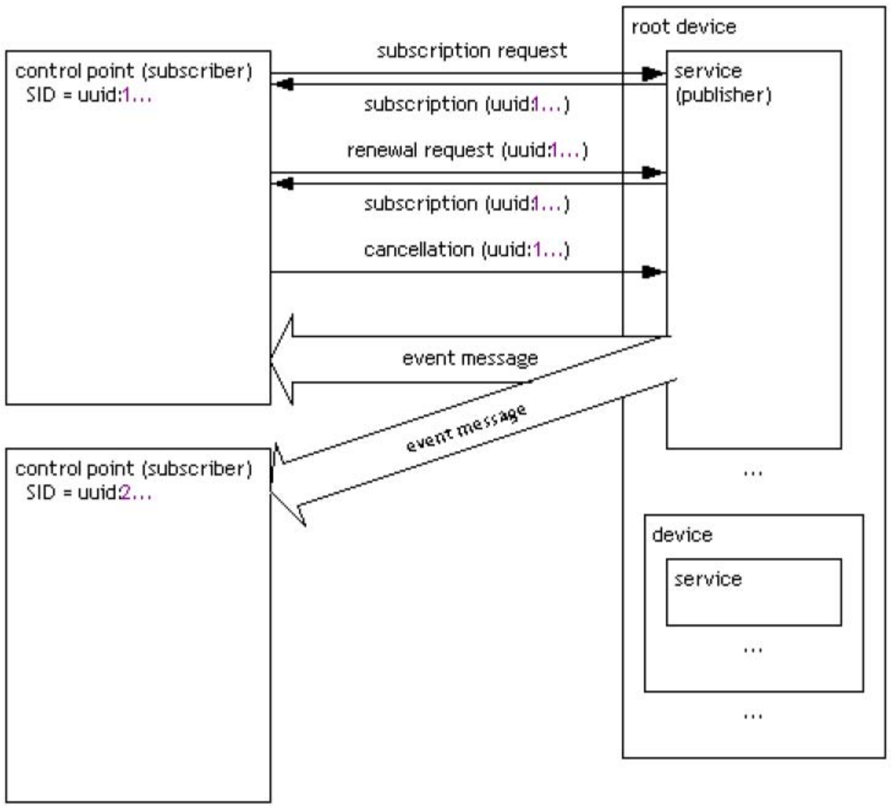


Figure 3.21: Eventing Process

To subscribe to eventing, a subscriber sends a subscription message. If the subscription is accepted, the publisher responds with duration for the subscription. To keep the subscription active, a subscriber must renew its subscription before the subscription expires. When a subscriber no longer needs eventing from a publisher, the subscriber should cancel its subscription.

A subscription is a request to receive all state variable changes. UPnP provides no mechanism to subscribe to events on a variable-by-variable basis. If a service accepts the subscription request, that service responds with a unique subscription identifier and the duration for the subscription. The unique identifier allows the CP to refer to the subscription in subsequent requests to the service, such as renewing or canceling the subscription. The duration specifies the length of time that the subscription is valid and maintained by the service. If one or more of a service's state variables are evented, the service publishes updates to the subscribers when the variables change.

To subscribe to eventing for a service, a subscriber must send a request with method SUBSCRIBE and NT and CALLBACK headers in the following format. Values in italics are placeholders for actual values.

```
SUBSCRIBE publisher path HTTP/1.1
HOST: publisher host:publisher port
CALLBACK: <delivery URL>
NT: upnp:event
TIMEOUT: Second-requested subscription duration
```
(No body for request with method SUBSCRIBE, but note that the message must have a blank line following the last HTTP header.)

If there are enough resources to maintain the subscription, the publisher should accept it. To accept the subscription, the publisher assigns a unique identifier for the subscription, assigns duration for the subscription. To accept a subscription request, a publisher must send a response in the following format within 30 seconds, including expected transmission time from the service to the CP.

```
HTTP/1.1 200 OK
DATE: when response was generated
SERVER: OS/version UPnP/1.0 product/version
SID: uuid:subscription-UUID
TIMEOUT: Second-actual subscription duration
```

If a publisher cannot accept the subscription, or if there is an error with the subscription request, the publisher must send a response with error messages (Incompatible headers, missing or invalid CALLBACK, Invalid NT or Unable to accept subscription). The response must be sent within 30 seconds, including expected transmission time.

As soon as possible after the publisher accepts the subscription, the publisher sends the first or initial event message to the subscriber. This message includes the names and current values for all evented variables. This is done as a convenience to allow the CP to initialize its representation of device's state. The CP doesn't have to explicitly request each variable or wait for an event to occur before it can receive the current value of all the device's evented state variables. This initial event message is always sent, even if the CP unsubscribes before it is delivered. The device must insure that the CP has received the response to the subscription request before sending the initial event message, to insure that the CP has received the SID (subscription ID) and can thereby correlate the event message to the subscription.

All subscriptions must be renewed periodically for the CPs to continue to receive notifications. To keep a subscription active, a CP must send a renewal message before that subscription expires. The renewal message is sent to the same URL as the original subscription message, but the renewal message does not include a delivery URL for event messages. Instead, the renewal message includes the subscription identifier received in the initial message accepting the subscription. When a CP no longer wants to receive events from a service, the CP can cancel its subscription by sending an appropriate message to the subscription URL. To renew a subscription to eventing for a service, a subscriber must send a request with method SUBSCRIBE and SID header in the following format.

```
SUBSCRIBE publisher path HTTP/1.1
HOST: publisher host:publisher port
SID: uuid:subscription UUID
TIMEOUT: Second-requested subscription duration
(blank line)
```

To accept a renewal, the publisher reassigns a duration for the subscription and must send a response in the same format and with the same conditions as a response to a request for a new subscription, except that the initial event message is not sent again. If a publisher cannot accept the renewal, or if there is an error with the renewal request, the publisher must send an error response (Incompatible headers, Missing SID or invalid SID, unable to accept renewal).

When a subscription expires, the subscription identifier becomes invalid, and the publisher stops sending event messages to that subscriber. If the subscriber tries to send any message other than a subscription request message - a subscription renewal request or a subscription cancellation message - the publisher rejects that message due to the invalid subscription identifier.

When eventing is no longer needed from a particular service, a cancellation message should be sent to that service's eventing URL. If a CP is removed abruptly from the network, it might be impossible to send a cancellation message. As a fallback, the subscription will eventually expire on its own unless renewed. To cancel a subscription to eventing for a service, a subscriber should send a request with method UNSUBSCRIBE in the following format.

```
UNSUBSCRIBE publisher path HTTP/1.1
HOST: publisher host:publisher port
SID: uuid:subscription UUID
(blank line)
```
(No body for request with method UNSUBSCRIBE)

To cancel a subscription, a publisher must send a response in the following format within 30 seconds, including expected transmission time. If there is an error with the cancellation request, the publisher must send a response with errors (Incompatible headers, Missing SID or invalid SID). The response must be sent within 30 seconds, including expected transmission time.

```
HTTP/1.1 200 OK
```

When an evented state variable changes, the service notifies subscribed CPs by sending event messages. These messages are GENA NOTIFY messages, sent using HTTP, that contain within their body an XML structure that specifies the names of one or more state variables and the new values of those variables. Event messages are sent as soon as possible after the state changes so that CPs receive accurate and timely information about the service, allowing them to display a responsive user interface.

```
NOTIFY delivery path HTTP/1.1
HOST: delivery host:delivery port
CONTENT-TYPE: text/xml
CONTENT-LENGTH: Bytes in body
NT: upnp:event
NTS: upnp:propchange
SID: uuid:subscription-UUID
SEQ: event key<?xml version="1.0"?>
<e:propertyset xmlns:e="urn:schemas-upnp-org:event-1-0">
   <e:property>
     <variableName>new value</variableName>
   </e:property>
Other variable names and values (if any) go here.
</e:propertyset>
```

Some state variable values might change too rapidly for eventing to be useful. Event notifications for such state variables may be filtered; devices send notifications only when the degree of change exceeds a limit or when a specified amount of time has elapsed since the last notification. The parameters for these filters are specified in the service description.

Figure 3.22 shows how a CP communicates with a service to subscribe to and receive notifications of state variable changes.

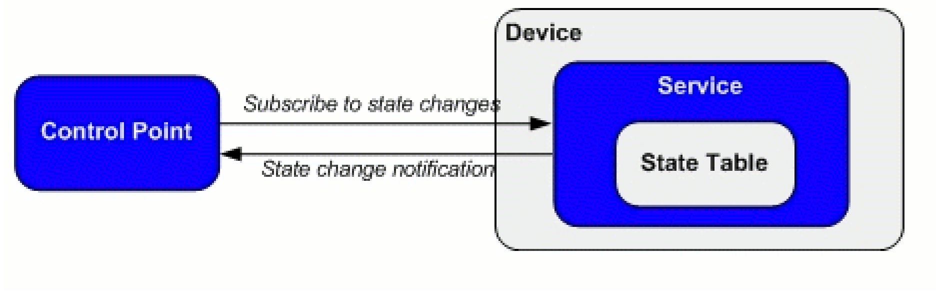


Figure 3.22: State Changes and Notifications

All subscribers are sent all event messages. Subscribers receive event messages for all evented variables, and event messages are sent regardless of the reason the state variable changed, e.g., in response to an action request or due to an internal state change.

In order to allow detection of lost or out-of-order messages, each event message has a sequence number or an event key having 32-bit integer value by the publisher for each subscription. The event key for a subscription is initialized to 0 when the publisher sends the initial event message. For each subsequent event message, the publisher increments the event key for a subscription, and includes that updated key in the event message. Overflow of that value must be handled by both publishers and subscribers. The publishing service is responsible for handling overflow by wrapping the event key back to 1 (not 0 - the value is reserved for the initial event message) while the subscribers must handle this special case by not interpreting it as an error.

To repair an event subscription, e.g., if a subscriber has missed one or more event messages, a subscriber must unsubscribe and re-subscribe. By doing so, the subscriber will get a new subscription identifier, a new initial event message, and a new event key.

CPs acknowledge receipt of event messages by responding with an “HTTP OK” message. In case of no response from a subscriber to any event message, the publisher should continue to attempt to send subsequent event messages to the subscriber until the subscription expires. If there is an error with the event message, the subscriber must respond with an error massage (Missing SID, Invalid SID, Missing NT or NTS header, Invalid NT header, Invalid NTS header). The response must be sent within 30 seconds, including expected transmission time.

3.2.4.6 Presentation

Presentation is the process where a device presents a browser-based user interface for manual user control and to allow viewing of that device's status. Each UPnP device contains an internal HTTP Web server and may provide a Web page for browser-based clients. That Web page serves as the device's manual interface that complements the device's programmatic SOAP-based control interface. The protocol for retrieving the presentation document, as with the description document, is HTTP over TCP. Retrieving a presentation page is a simple HTTP-based process and uses the following subset of the overall UPnP protocol stack in Figure 3.23.

UPnP vendor
UPnP Device Architecture
HTTP
TCP
IP

Figure 3.23: UPnP Presentation Protocol Stack

The CP can use the presentation URL contained in the description document to request the presentation document. Not all devices have a presentation document nor are all CPs able to display a presentation document containing complex HTML objects such as frames, embedded Java applets, and so on. The browser-based interface is used to control the device, to change operational parameters, to view device and service information, or for any other device-specific functionality implemented by the manufacturer. The presentation page provides a simple, consistent way to work with UPnP devices and requires no custom software. The degree to which control and status monitoring can be accomplished depends on the specific capabilities of the presentation page and device. UPnP Presentation phase is shown in Figure 3.24.

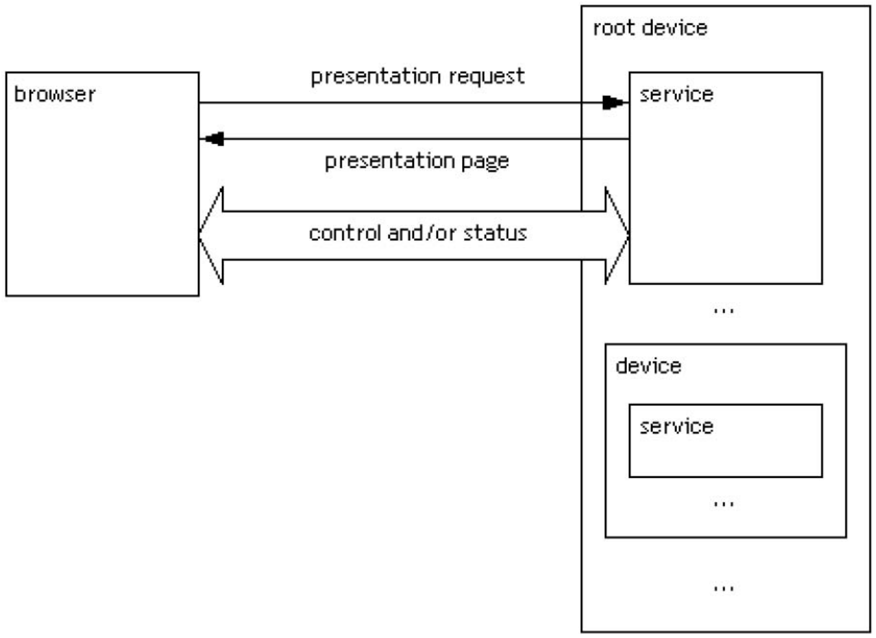


Figure 3.24: UPnP Presentation Phase

Unlike the UPnP device and service descriptions, the contents of a presentation page are completely specified by the UPnP vendor. A presentation page need not be present, but if present it must be an HTML page and should be written in HTML version 3.0 or later.

3.3 UPnP Technology's Relation to Other Technologies which are Alternate Standards for Home Networking

Networking Media for UPnP leverages the standard IP protocol suite to remain network media agnostic. Devices on an UPnP network can be connected using any communications media including Radio Frequency (RF, wireless), phone line, power line, IrDA, Ethernet, and IEEE 1394. In other words, any medium that can be used to network devices together can enable UPnP. The only concern might be that the media being used supports the bandwidth required for the intended use. UPnP uses open, standard protocols such as TCP/IP, HTTP and XML. However, other technologies could be used to network devices together for many reasons, including cost, technology requirements, or legacy support. These include networking technologies like HAVI, CeBus, LonWorks or X10. These too can participate in the UPnP network through an UPnP bridge.

TCP/IP is a great foundation for home networking. Today, there are many different types of networks, even within a home – security, telephone, HVAC, intercom, cable TV, etc. These individual networks are effectively islands that in most cases do not interoperate with one another. One big advantage of TCP/IP is that many applications and services can be supported over one network. In fact, these various other networking types can be bridged to work with applications and services running on TCP/IP for even more choice and innovation while enabling a user to continue benefiting from legacy, installed networks. An UPnP network containing bridged devices might look like the Figure 3.25 below.

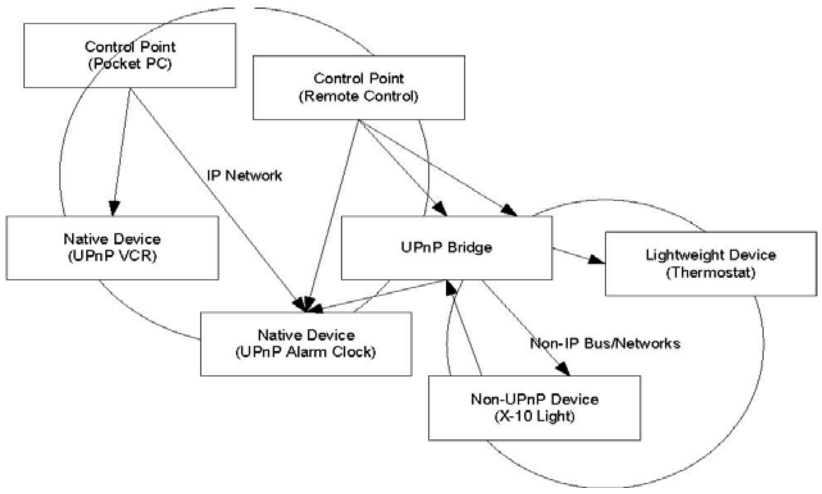


Figure 3.25: Bridged UPnP Network

There are some proprietary, vendor-specific home networking solutions that run over TCP/IP-based networks. Since it embraces an Internet standards-based approach, UPnP technology supports a full range of applications and services rather than just one or more isolated applications or services, as such proprietary approaches support. This enables UPnP technology to offer consumers more simplicity, choice and innovation while providing vendors a flexible, economic foundation for various products and service.

UPnP Technology and OSGI are complementary technologies [22]. UPnP technology is focused on enabling the automatic discovery of various devices and services across a home network based on existing Internet standards. OSGI specifications are focused on enabling the delivery of managed broadband services to networks in homes, cars and other environments. UPnP technology and OSGI address different problem sets and are complementary. In fact, some companies have already introduced products that incorporate both UPnP technology and OSGI specifications.

UPnP technology and IEEE 1394 are complementary technologies [6]. IEEE 1394 is a network technology, not unlike Ethernet, and can support TCP/IP running over it. Because UPnP technology can operate over any type of TCP/IP network medium, UPnP solutions can operate over a 1394 network.

The UPnP architecture specifies a language- and network-independent protocol while X-10 is a physical layer networking medium running over a home's power lines. X-10 is not capable of implementing UPnP within the individual end devices, but it is conceivable manufacturers could implement UPnP within the basic controlling device and that device would be UPnP-capable.

UPnP technology and Home Audio Video Interoperability (HAVI) [21] offers an integrated solution for audio/video communication on IEEE 1394 networks, while UPnP technology provides a general solution for networks using Internet Protocol. HAVI and UPnP can be bridged.

UPnP permits the use of proxies for devices and subsystems that can't imbed the UPnP layer. For example, an intelligent light switch can not be expected to contain an IP stack, an Ethernet controller, and the UPnP layer. A LonWorks-to-UPnP proxy allows the UPnP space to communicate with the control devices on the LonWorks network. It also lets the LonWorks devices access the UPnP space and devices.

Simple Control Protocol (SCP) is a very lightweight device control protocol that allows manufacturers to create small, intelligent devices that can communicate with each other, securely and robustly, over low-speed communication networks [25]. SCP devices use the same device models and schemas developed by the Universal Plug and Play (UPnP) Forum, allowing SCP and UPnP devices to interoperate easily through a bridge. This extends the full capabilities of UPnP into the world of small devices which can't afford the cost of a TCP/IP and native UPnP stacks. For example, SCP device properties can be set through messages sent across the Internet. All SCP device property relationships are established at the UPnP level through the Bridge and propagate down to the SCP network and device level. UPnP bridging also allows SCP devices to interoperate with devices that use other control protocols. If a Bridge exists between those devices and an UPnP network, SCP devices can use UPnP as a common communication language.

Simple UPnP Proxy Protocol (SUPP) is the protocol to communicate with a proxy, which extends the device with Universal Plug and Play [24]. SUPP maps control and eventing of UPnP to a simple and lightweight communication protocol. Thus it is possible to design device independent UPnP proxy modules, chips or applications, to enable devices with UPnP networking.

With SCP**,** a device or a CP with little hard- and software resources and performance to implement Universal Plug and Play, a UPnP Proxy in the form of a device independent network module, a network chip or another component can take over the UPnP functionality. By defining a standardized protocol for the communication between the device and the UPnP Proxy at a high level, there are many combinations of physical network and device interfaces possible.

4. SECURITY IN UPNP

UPnP version 1 does not directly specify any counter measures for security threats. So, connect to an UPnP device network and listen to messages or control and query UPnP devices for an unknown and potentially hostile entity is possible. It instead leverages security features found in the standards-based protocols upon which the UPnP architecture is built. Among these are network isolation (firewalls), MAC layer encryption and physical access control [26]. Because these measures alone are not sufficient to address all scenarios, a security working committee has been formed within the Forum to investigate potential future security enhancements.

The UPnP Forum published a security extension that allows the equipments of the network to exchange messages with some proof or kind of assurance on the identities of the sender and the addressee, and possibly by ciphering contents.

UPnP Security protects UPnP messages from tampering and/or disclosure to not trusted parties, and it protects UPnP devices from unauthorized control operations [29].

UPnP Security defines a service to be added to each secured device that allows its security to be managed. It also defines a service and CP behavior for an application called a SC, which edits the Access Control List (ACL) of a secured UPnP device and controls other security functions of that Device. UPnP Security also permits fine-grained access control so that some CPs can be granted limited access. UPnP Security uses public-key cryptography to identify principals (entities to whom access is granted). However, unlike secure Web server public-key certificates, the public keys used in UPnP Security do not need to be issued or certified by a centralized authority. The decision to trust a public key is made by the device owner during device and/or CP installation.

Another important aspect of UPnP access control is the set of permissions that can be granted. The approach taken by UPnP Security is to define access control in terms of device-specific sets of abstract permissions. Each permission potentially enables one or more actions to be performed by a CP authorized by an entry in the device’s Access Control List (ACL). Device manufacturers are free to define their own permissions and use whatever mechanism they want to map incoming control requests onto required permissions.

In UPnP, there are devices and CPs. Devices advertise themselves via a discovery protocol and offer services: collections of SOAP actions that the CPs invoke.

Three protocols used in UPnP architecture (SOAP, SSDP and GENA) to let components interact with each other are secured with the following ways:

UPnP Security secures SOAP control messages and their replies. This message security consists of, identification, integrity, authentication, freshness, authorization and secrecy. Control message security is depicted in the flowchart of Figure 4.1.

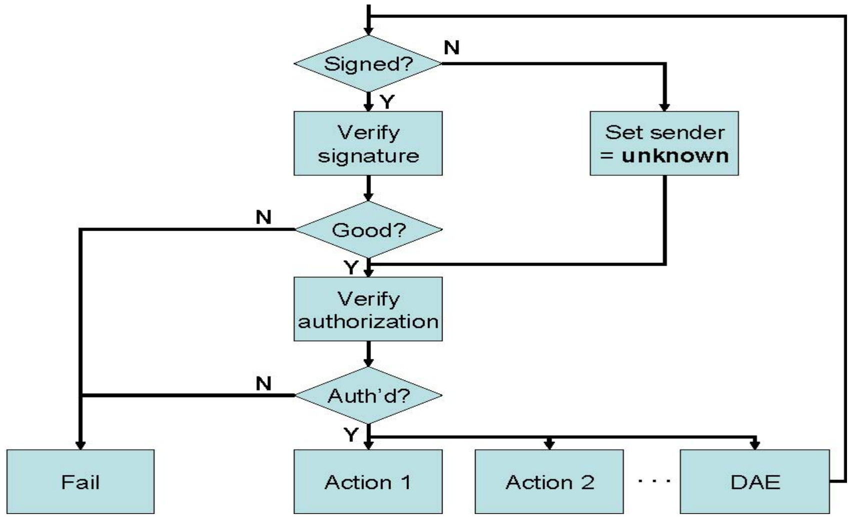


Figure 4.1: Control Message Security Flowcharts

1. A message arrives and is parsed. If there is a <SecurityInfo> element in the SOAP header, then this message is assumed to be signed.

2. If the message is not signed, then set its owner to “unknown” and proceed with testing for authorization. (Some actions are made available to unknown senders, at the discretion of the device working committee, device manufacturer or device owner.)

3. If the message is signed, then the device verifies its signature and its freshness. Verification of the signature verifies both integrity of the message and origin authentication. The sender is indicated by its SecurityID – the hash of its public key.

4. If the signature and freshness are good, then the owner is set to the SecurityID of the signer of the message. Otherwise, the message fails and an error return message is generated.

5. If the message is to be tested for authorization, then authorization information is gathered. This consists of the owner list, ACL, any cached certificates and any certificates provided in the message's <KeyInfo> element. The message's action name is retrieved and used to discover the abstract permission needed by the sender to perform this action. The authorization information of the sender is then tested to determine if that permission has been granted and is still in force.

6. If the sender is not authorized, then the message fails and an error return message is generated. If the sender is authorized, then the desired action is executed.

7. If a CP needs an action message to be confidential, it uses the DecryptAndExecute action (indicated in Figure 4.1 as "DAE"), then its cipher text argument is deciphered and the resulting plaintext is recursively sent back to the start of the process for processing. Usually, DecryptAndExecute will not be a signed message but the message inside its argument could be. The reply from that action is then encrypted and returned in the reply to the DecryptAndExecute action.

SOAP is secured by allowing only authorized CPs to invoke any secured action within a Device. This is accomplished by an ACL in each secured Device, each of the entries of which lists a CP unique ID, a name of a group of CPs or the universal group <any/>, and what that CP or group is allowed to do on that Device.

SSDP is difficult to secure. It is vital for the authorized user to discover other devices on the home network, but it is desirable that an attacker not be able to take an inventory of the home network's equipment. Therefore, to secure SSDP, the existence of devices is announced, but they are announced as generic devices, and are not described in any detail except in response to requests from authorized CPs. It is still possible for a determined attacker to take an inventory of a home network, even if all traffic was completely encrypted (e.g., via IPSEC), just based on timing and length of messages. Therefore, securing SSDP is considered low priority until there are significant new research results in anonymity protection.

For security of GENA, a normal event variable is defined that anyone can read, named "EncryptedEvent" but it contains any event variables that are to be made available only to authorized CPs. An EncryptedEvent is sent to a CP encrypted in a key known only to the device and that CP. CPs not allowed to see such an event, are not allowed to subscribe to those events, and are not able to see the events by eavesdropping on the network.

4.1 UPnP Security Actors

There are several classes of actors that play a role in the UPnP security world: the owner, the SC(s) and the security-aware UPnP devices and CPs as shown in Figure 4.2 [29].

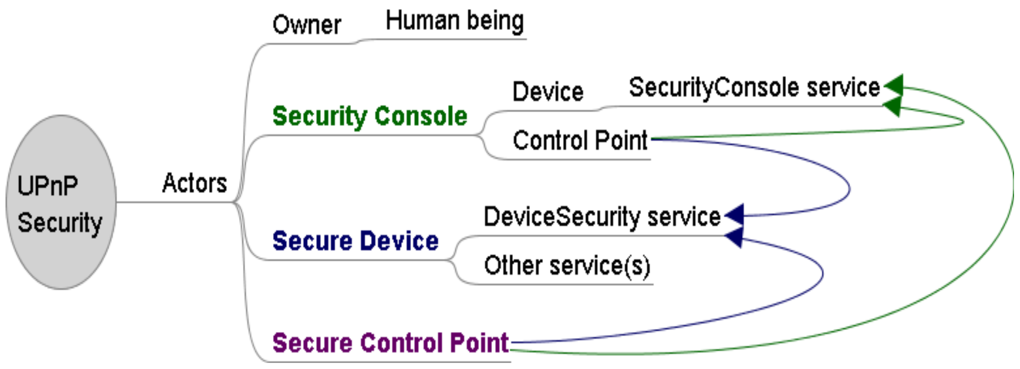


Figure 4.2: UPnP Security Actors

- The owner is any human operator who is knowledgeable enough to operate the devices and to activate their security features.
- From the point of view of UPnP, the SC is a user application that is both a device and a CP. Its function is to give the user an interface by which to administer access control on the user's own devices [28]. As part of that function, the SC maintains a list of all devices, CPs and other SCs that constitute the user's "own" domain. The SC enables the user to define that set manually. Its purpose is to take security ownership of remote secure devices (through its DeviceSecurity service) and then to authorize CPs (or other SCs) to have access to Devices over which the SC has control.

  A SC can own a Device, meaning that it has the right to edit that Device's ACL and can do anything on that Device, or it can have been given some subset of rights to the Device and have the privilege to grant all or some of those rights to another SC or CP. Sometimes, the grant of authorization by a SC to a CP (or other SC) cannot be achieved by ACL editing. In case of a device might not have memory to hold any more ACL entries or the SC doing the grant of authorization might not have been granted ACL editing permission or the device might not be online. In those circumstances, the grant of authorization must be by authorization certificate. These certificates must be communicated to the grantee (or to the affected device, if it has room in its certificate cache). The SC also has the job of defining names of individual CPs and of groups of CPs and offers a user interface for administration of access control on security-aware UPnP devices.

Also, as a CP, each SC must announce its own keys to any other SC it discovers, by use of PresentKey. However, a SC is also a device.

As a device it is self-owned. If it has any access controlled actions, then those are to be administered by the human user and not by some other SC. Therefore, a SC does not need to include a DeviceSecurity service.

- Security-aware devices are UPnP devices which implement the DeviceSecurity service [27]. Since this service does not provide any end-user oriented feature, it is highly expected that other, more common services will also be found. The actions provided by the service are divided into two groups. One is meant to be used only by SCs; the other can be used by both SCs and CPs.
- Security-aware CPs are UPnP CPs that know how to handle the DeviceSecurity service of a secure device. Indeed they need to be able to negotiate a session key to ensure integrity and/or confidentiality of the messages. These CPs must also look for devices implementing the SecurityConsole service. This is the only way for them to announce themselves as secure CP to the corresponding SCs. A SC doesn't need and cannot differentiate a simple security-aware CP from another SC; both kinds are handled in the same way.

4.2 Keys

To have secure communications, both symmetric and asymmetric cryptographic means are specified and used. “Public Key” is a short-cut to introduce “asymmetric cryptography based on a pair of public and private keys” [26].

Asymmetric keys are used during the identification phase to provide some assurance that a given remote device is really the one that the end-user wants to use and not an attacker trying to impersonate the device. Asymmetric keys are also used to establish secure sessions. So far, the only algorithm that has been standardized is RSA with 1024-bit keys.

Once the identification of the device is over, a usual way to reduce the workload of the processor is to encrypt the sessions with symmetric keys (128-bit AES is standardized). Session signatures are also computed with symmetric algorithm (SHA1 HMAC), even though there is a room left for asymmetric signatures, using for instance the RSA signature algorithm.

4.2.1 Exchanging Keys

Every SCs and devices own a pair of public and private keys. If a UPnP equipment is both a secure device and a secure CP it has two pairs of keys.

- The public key of a device can be retrieved by a CP through a call to the GetPublicKeys action. This action can be executed by any CP without access control since the provided key is public. The obtained key is then used by the CP to encrypt the call to the SetSessionKeys action. Note that for the time being, according to UPnP Security devices are only expected to use confidentiality keys (that is a key to encrypt messages, not to sign them).
- The public key of a CP is provided to a SC by calling the PresentKey action of the SecurityConsole service. A “preferred name” can be provided at the same time to help the end-user administrate its CPs. This default name is especially useful when more than one CP were discovered and they would otherwise be listed with cryptic, unfriendly names.
- UPnP Security assigns a more important role to the SC that is first to connect on behalf of the end-user to a secure device and take control of it, compared to the following ones. The public key of this first SC is provided to the device by calling the TakeOwnership method. The public keys of the following SCs are provided by calling the GrantOwnership action. Finally, since these actions can be automatically executed when new equipment is discovered, no special right must be granted before the end-user validates the legitimacy of the newcomer.

4.2.2 Keys Usage

As one has already noticed, the relationship between CPs and devices is not symmetric: CPs choose their devices and always are the initiators of any exchange of messages. To ensure confidentiality, integrity, non-replay of the exchanges, the following rules are defined:

- CPs (and SCs) use their public keys to sign messages. Signing is mandatory to warrant that no equipment can impersonate a legitimate, registered CP and abuse a secure device.

On the other hand CPs need not provide asymmetric keys for encryption. Indeed once a symmetric session key has been established; it is much more efficient and less resource consuming.

- The public key of a device is used to encrypt the call to the SetSessionKeys actions. All subsequent messages use the defined symmetric key.
- A CP that wants to conduct a control session with a security aware device may call GetAlgorithmsAndProtocols, in order to confirm interoperability, and will then call SetSessionKeys, to establish a secure session with the device. These sessions do not hold network connections open and could be very long-lived, depending on the storage capacity of the Device and the CP. Once a CP has established a session with the device, it invokes actions by sending normal action messages, digitally signed using XML-Signature with a symmetric signature key (e.g., HMAC) established during SetSessionKeys and with a sequence number initialized by that action. The sequence numbers during a session must be monotonically increasing but need not be sequential. A session may be ended intentionally, by ExpireSessionKeys, or may time out at the device's discretion.

4.3 Discovery and Component Naming

The first security requirement is to discover your own network components. As shown in Figure 4.3, UPnP uses a broadcast protocol, SSDP, for physical discovery of Devices but nothing for discovery of CPs. Therefore, the SC acting as a Device, offers an action (PresentKey) by which security-capable CPs can announce themselves to the SC. In that way, it learns of local CPs while it uses SSDP to learn of local Devices.

Discovery of a secure device has two phases. The first phase is used by the SC to get the public key of the device and let the end-user associate a friendly name with the corresponding Security ID. To be sure that no pirate device tries to impersonate the real one, the end-user is asked to verify that the security ID computed by the SC matches the ID directly provided by the device. During the second phase, the security ID of the SC is uploaded into the device memory. The risk at this moment is that a malicious SC tries to take control of the device in place of the end-user's real one. To prevent this, the owner-to-be SC must provide a password that only a human being that has a physical access to the device can know.

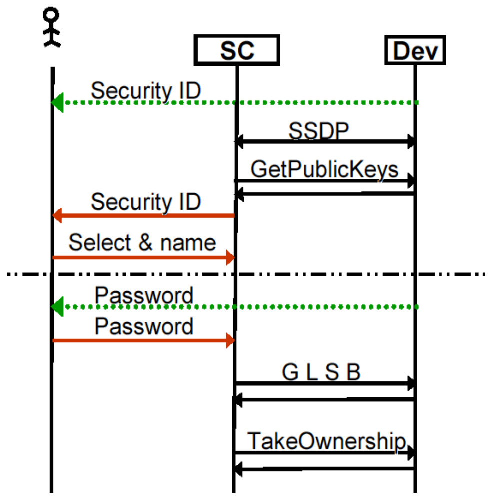


Figure 4.3: Device Discovery Ceremony and TakeOwnership

GLSB stands for GetLifetimeSequenceBase and is an action that returns a different value each time to ensure that the TakeOwnership exchange cannot be replayed by an attacker. After these two steps, both parties are sure that the remote peer is not faked and can thus be trusted.

In pre-security UPnP, there were no components that would need a list of CPs and CPs could be completely anonymous. With UPnP Security, the user must grant access rights to CPs and must therefore both list and name them. The discovery of the list of CPs and the assignment of names to them is performed by the SC. In Figure 4.4, rather than change the nature of CPs, by having them engage in SSDP advertising, the SC advertises itself and offers an action PresentKey, by which a security-aware CP announces itself to the SC.

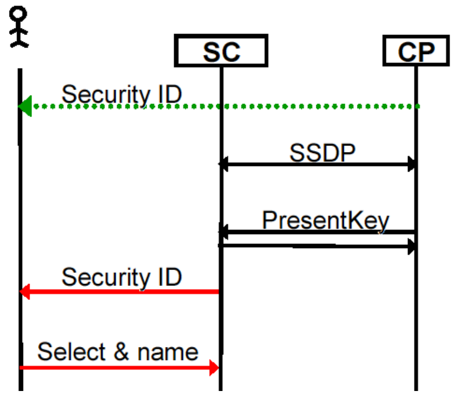


Figure 4.4: CP Discovery

Black arrows are used for the network communication. Red arrows are for normal user interface and green dotted arrow are for static user interface (like reading a label on a device).

Selection of one's own components can be done in a variety of ways, and there is room for much creativity. UPnP Security offers a generic method, using a Security ID for each component. The Security ID is the SHA-1 (Secure Hash Algorithm) hash of a component's public key, expressed in BASE32 (using only upper case letters and six of the ten digits). It looks like a product registration ID:

GM3GK-RTMOI-4GYK2-ZK5FC-WMTRK

This ID is unique among all components in the world. By comparing the Security ID of a physical component to the ID announced to the SC, one can determine precisely which Devices are his or hers, even if the local network contains hundreds of Devices of the same kind.

Once a Device is selected, whether physically or via comparison of Security IDs or otherwise, the user needs a better way to refer to the Device. The manufacturer could supply a name for the Device, but the user is given the chance to set his or her own name for the component.

That name needs to be meaningful only to that user, so the choice of name is entirely up to the user. The name is used by the SC user interface, while over the network the hash of the public key (from which the Security ID is generated) is used as the components unique ID. The key hash is also the ID by which a CP is known in a certificate or Device ACL. The Discovery and Component Naming Architecture is shown in Figure 4.5.

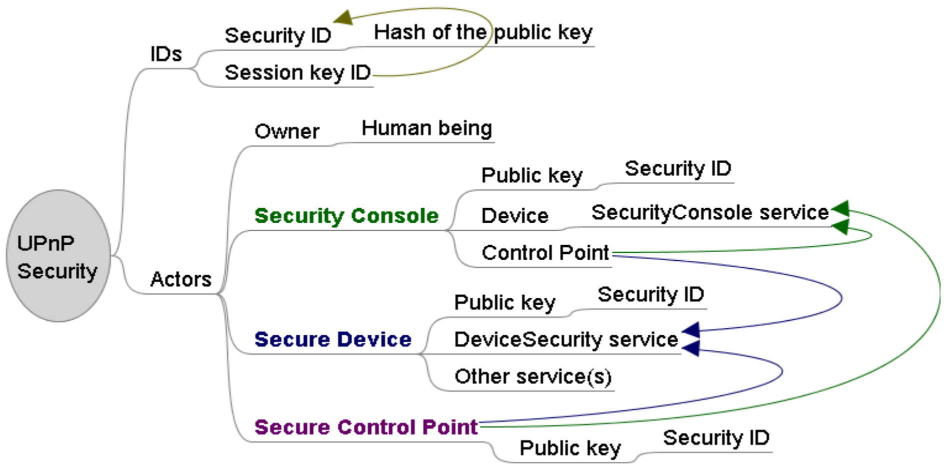


Figure 4.5: Discovery and Component Naming Architecture

4.4 Security Ownership

A CP does not need to be exclusive about which SCs it advertises itself to. It is the beneficiary of grants of authority and all decision making is done by the SC in that case.

The situation is reversed for Devices. A Device has the resources (SOAP Actions) to which access must be restricted. The SC, by editing the Device's ACL, tells the Device which CPs to obey. Therefore, the Device needs to be very exclusive in choosing which SC to associate with. This process is called "security ownership," in UPnP Security.

Every device that implements the DeviceSecurity service must have an owners list with at least one entry. This list contains all the SCs that are allowed to call every actions of the device, including defining the access rights of the other CPs.

By the generic ownership protocol defined by UPnP Security, an SC can take ownership of a Device only if the SC knows the Device's secret password and the Device is not already owned. Once a device is owned, an SC that owns it can grant co-ownership to another SC or revoke it, but more importantly, an SC that owns a Device can completely re-write the Device's ACL (or do any other ACL editing operation). When a device is removed from the network, the FactorySecurityReset action is called in order to blank its owners list.

4.5 Access Control

Each device is responsible for its own access control decisions. Those decisions are made according to a security policy, established in turn by the device's owner, through the actions of a SC. The security policy for a device is encoded in the form of three data structures, of decreasing power (of granted access) as shown in Figure 4.6.

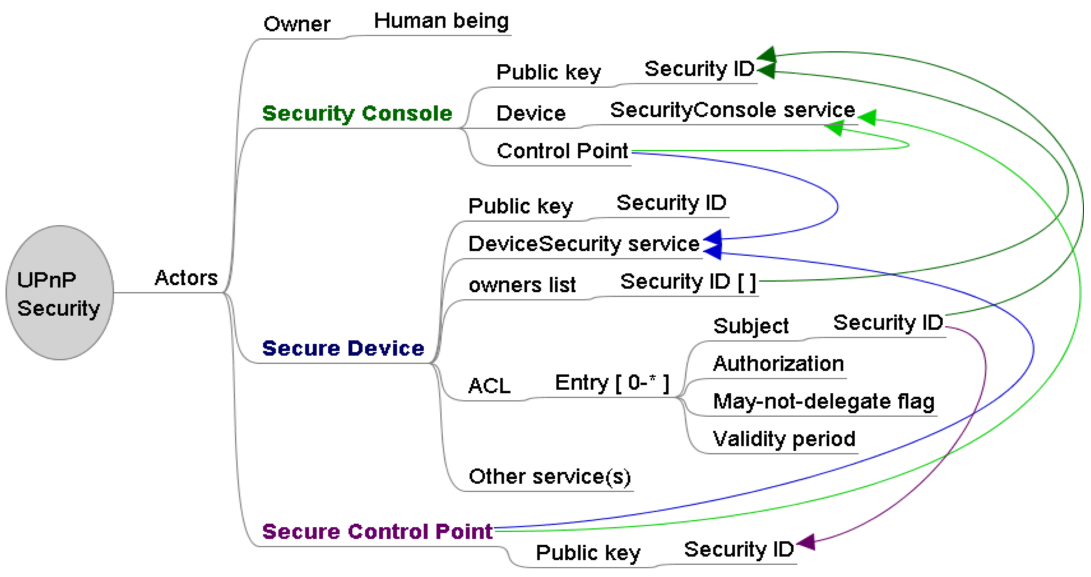


Figure 4.6: Security Policy of Device

4.5.1 List of Device Owners

The list of device owners is the list of (the hashes of) those signature keys that are permitted to edit the ACL of the device. By default, each of these keys is given total permission to operate the device as well. An owner is indicated by the hash of the public key of a SC. An owner list is a list of such hash values.

4.5.2 Access Control List (ACL)

A device's ACL is a list of entries (possibly empty) that specify the following:

- Subject: a SecurityID or name of a group of SecurityIDs – indicating the (single, group of) CP or SC to which rights are being granted by this ACL entry
- Authorization: a list of permission elements, specifying what the Subject is being granted by this entry
- May-not-delegate: an optional flag indicating that the Subject may not delegate any of the rights granted in this entry on to other CPs, SCs or groups
- Validity: an optional set of elements that could include a not-before date/time and/or a not-after date/time, limiting the period of validity of the entry

The most straight-forward method of granting access permission is through modification of the device's Access Control List (ACL). The Access Control List is the root of the security policy of each device. It can only be defined by the end-user because one needs to know which CPs are available on the network and for which purpose. The precise contents of the ACL and what is possible depend on the services offered by the device.

Each ACL entry lists a signature key (indicated by its hash) and one or more permissions being granted that key. These permissions are defined by the device manufacturer and correspond to some set of actions that the signature key is allowed to perform. The mapping from defined permission to set of permitted actions is left up to the manufacturer to specify. An ACL entry might also contain limiting specifications: validity date/time limits and, possibly, the prohibition against further delegation. These limitations are not expected in normal ACL entries, but they are available for advanced SC-Device interaction.

In some cases, it is not possible or not desirable to grant permission by modifying an ACL. These include:

- When the SC that is granting that permission (that is, is in communication with the CP that is being granted the permission), does not have permission to edit the ACL of the device
- When the granting SC is not in communication with the target device
- When the device does not have enough memory to hold any more ACL entries

In these cases, access is granted by authorization certificate. An authorization certificate can be thought of as a digitally signed ACL entry.

4.5.3 Authorization Certificates

Certificates must be communicated to the device that is doing the access control test. They can be communicated directly, via a CacheCertificate call to the device itself, but they will probably be communicated to the CP being empowered, for that CP to pass along to the device.

The CP can pass those certificates to the device via CacheCertificate or within an XMLSignature <ds:KeyInfo> element. The former has a performance advantage, in the case of repeated authorized actions, but may not be allowed due to device memory constraints. The communication of certificates from the SC to the CP is provided via the SecurityConsole:1 service.

There are two forms of certificates defined for UPnP Security: an authorization certificate and a named group membership certificate.

A named group membership certificate contains the following entries:

- Issuer: the SecurityID of the issuing SC
- Name: the textual name of the group
- Subject: a SecurityID or name (as in an ACL entry) of the individual key or group of keys being given membership in this group
- Validity: an optional list of elements that could include a not-before date/time, a not-after date/time and/or a <Renew/> element (noting that this certificate was issued to be valid for an artificially short interval of time and can be renewed by sending it to the Issuer).

Name definition certificate adds one key (or one whole group) to a named group. A key is added by referring to its hash. A named group is added by referring to its name.

An authorization certificate has the same elements as an ACL entry, but also an Issuer field and all the same validity options as the name certificate. The authorization field is slightly different because it must indicate the device(s) to which the grant of authorization applies while the ACL can apply only to the one device in which it is resident. A subject listed in the device ACL might be given the right to delegate some set of permissions. That subject can delegate permissions on to a second subject, where what gets delegated to the second subject is the intersection of the rights granted the first subject and the rights delegated on to the second. With delegation of rights, the burden of administering security is spread out. With delegation by authorization certificate, the entity to whom rights have been delegated does not get total rights to the device, cannot further delegate any more than the rights it has been given to delegate, and cannot remove rights of others.

4.6 The Logic of Operation from the Security-Aware Device's Point Of View

- If a device has a display or printing capability and also has a source of randomness, then it is preferable for the device to generate a new password and new public key pair on any power-up when it is in factory reset state. It would then display or print the generated password and the hash of the new public key. For devices without those abilities, the password and public key pair will have been created by the manufacturer and will remain constant over the lifetime of the device. In that case, the manufacturer will have printed the password and the Security ID (hash of the device's public key) on a card or a label attached to the device case, or both. The password does not need to be longer than about 6 upper case alphanumeric characters, provided it was generated randomly for that device and is used by TakeOwnership shortly after the device is plugged into the network. [Note: passwords or keys are not to be used in common across a set of devices. Such usage would be a security flaw.]

- The device announces itself via SSDP, perhaps giving real details or perhaps describing itself only as "Security Aware Device", if the device wants to prevent inventory by an unauthorized CP.
- A SC, presumably that of the device's owner, calls the GetPublicKeys, GetLifetimeSequenceBase and TakeOwnership actions, supplying the device's password to prove authorization to take control of the device. As a result of a successful TakeOwnership action, that SC is listed as the device's Owner. An Owner is a CP that is empowered to edit the device's Access Control List (ACL). Subsequent TakeOwnership attempts MUST be ignored. If the device generated its password dynamically, then the password used in this TakeOwnership action should not be valid again.
- The owning SC establishes a set of session keys by calling GetLifetimeSequenceBase and SetSessionKeys. These session keys will be used for digitally signing future action messages using XML-Signature with symmetric key signature.
- The owning SC will then be free to use the GetACLSizes action to discover whether access permissions can be granted by ACL entries directly or need to be encoded as certificates, because the device has too little room to store an ACL.
- The owning SC will probably also invoke GetDefinedPermissions in order to learn what permissions it might grant to chosen CPs.
- If the SC grants access by certificate, that operation happens without invoking any actions on the device. Otherwise, the SC grants access to desired CPs by way of AddACLEntry. It also reads the current ACL contents, for display to its human operator, by using ReadACL. The actions WriteACL, ReplaceACLEntry and DeleteACLEntry are also available for ACL editing. These actions will be omitted from DeviceSecurity in those implementations that do not provide any memory for an ACL.
- A CP that wants to conduct a control session with a security aware device may call GetAlgorithmsAndProtocols, in order to confirm interoperability, and will then call

SetSessionKeys, to establish a secure session with the device. These sessions do not hold network connections open and could be very long-lived, depending on the storage capacity of the Device and the CP.

- Once a CP has established a session with the device, it invokes actions by sending normal action messages, digitally signed using XML-Signature with a symmetric signature key (e.g., HMAC) established during SetSessionKeys and with a sequence number initialized by that action. The sequence numbers during a session must be monotonically increasing but need not be sequential.
- If a CP needs an action message to be confidential, it uses the DecryptAndExecute action, one argument of which is the cipher text of an encrypted action. The reply from that action is then encrypted and returned in the reply to the DecryptAndExecute action.
- A session may be ended intentionally, by ExpireSessionKeys, or may time out at the device's discretion.
- In the event that the device's owner wants to share ownership, either with another person or (for fault-tolerance) with another SC operated by him- or her-self, a current owning SC can invoke GrantOwnership. This assumes that GetACLSizes shows that there is room in the device to record the additional owner. Ownership can be revoked via RevokeOwnership. Current owners can be listed via the action ListOwners.

Enhancements

- A device may, if it chooses, define named sets of permissions, called Profiles, and a SC may read those definitions via GetDefinedProfiles. These Profiles could be role names, for example, like Parent, Child, or Administrator. The Profile names can be used in the user interface provided by the SC.
- If a CP has certificates to present to the device, in order to gain access, and if the device shows via GetACLSizes that it has certificate cache memory available, the CP may send those certificates to the device via the CacheCertificate action. Such certificates would then be available on the device over multiple subsequent actions. Implementation of certificate caches is up to the device, but it would make sense to validate certificates at the time they are cached. It might also make sense to derive implied ACL entries from validated certificates and current ACL entries, and store those derived entries at the time a certificate is cached.

5. IMPROVEMENTS IN UPNP WITH THE UPnP 1.1 DEVICE ARCHITECTURE

5.1 Improvements in UPnP with the UPnP 1.1 Device Architecture

The UPnP 1.0 peer-2-peer home networking architecture is an important step towards interoperability of networked consumer devices, realizing the connected home experience.

The UPnP Forum initiated the design of the UPnP 2.0 device architecture in 2002, to address two main issues of the UPnP 1.0 architecture:

- It was based on drafts. Some drafts had expired; others like SOAP 1.1 had evolved into standards that are similar yet incompatible.
- Its SSDP has peak loads and is unreliable over wireless networks.

Unfortunately, the UPnP 2.0 device architecture was also designed around still evolving Web Services drafts. Rather than risking incompatibility and market fragmentation, in 2003 it was decided to create a backwards-compatible version: the UPnP 1.1 device architecture. This architecture should improve SSDP and adhere to external standards, without creating new dependencies on drafts.

UPnP 1.1 device architecture improves the UPnP 1.0 device architecture in a backwards-compatible way, improving discovery and compatibility with referenced standards [30]. The work on the UPnP 1.1 device architecture has shown that one can maintain backwards compatibility and innovate. Goals of the design were: improved performance and reliability, elimination of ambiguities and better interoperability, and compatibility with external standards to simplify implementation efforts and to simplify adoption of new technologies in the device architecture.

5.1.1 IPv6 Support

The UPnP 1.0 IPv6 annex does not define IPv6 address assignment. IPv6 support in the UPnP 1.1 device architecture normatively requires support for IPv4, link-local addresses and stateless auto configuration while DHCPv6 is optional.

UPnP 1.0 IPv6 annex define use of site-local addresses and site-local scopes not a global IPv6 addresses. The UPnP 1.1 IPv6 annex deprecates site-local addresses in favor of the well-standardized IPv6 global addresses.

IPv6 defines link local scope FF02, site local scope FF05 and global scope FF0E [31]. The smaller scopes that should be used for discovery announcements and searches, limit propagation of multicasts. This feature that is not available in IPv4. In larger networks, announcing on-site scope could potentially be useful for global addresses (even though site-local addresses are deprecated by UPnP 1.1): routers can forward site scope announcements depending on the source address which results in a lower overhead than announcing and searching on global scope. The use of both global scope with default hop limit 2 and site-local scope (no hop limit) were decided.

5.1.2 Discovery

The UPnP 1.1 device architecture revamped the SSDP, with five major performance and reliability improvements:

5.1.2.1 Spreading of Alive Messages

Spreading announcement messages over the entire interval to avoid peak network loads and to lower network overhead is recommended in the UPnP 1.1 device architecture. Some UPnP 1.0 CP implementations only listen for a specific message. To ensure that such CPs are quickly notified of a new device, the initial set of announcement messages is still sent as a burst. Subsequent announcement messages are spread over the entire interval. This is backwards compatible for two reasons. First, the UPnP 1.0 device architecture does not require burst behavior, so CPs cannot rely on known timing of bursts. Second, CPs need to deal with packet losses; hence they can already deal with only a subset of all announcements arriving.

Typical behavior of UPnP 1.0 implementations versus the newly recommended behavior is symbolized in Figure 5.1. It can be seen that the spread behavior avoids peak loads, and with the same average load the interval between announcements is much reduced. This will reduce the impact of burst losses and can have a low average load (lengthen the period of refreshments) with shorter intervals between messages.

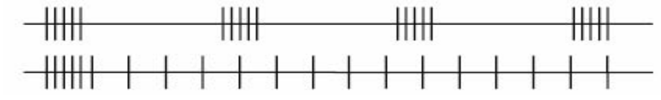


Figure 5.1: Bursts versus Spreading

5.1.2.2 All-or-nothing "Alive" Rules

The UPnP 1.0 "alive" rules specify that advertisements of a device and its services expire on their own, independent of the status of other advertisements. If some re-advertisement messages (or bye-bye messages) are lost while others are received, it is possible that a CP has "alive" services in a "dead" device in its cache (or the other way around). The UPnP 1.0 device architecture does not specify how a CP should treat such a device, and this confused state can lead to inconsistencies.

The UPnP 1.1 device architecture solves this problem by simplifying: a device and its services always have the same alive/dead state. Only if all advertisements have expired, or if at least one bye-bye message has been received does a CP consider a device and its services gone from the network.

5.1.2.3 Lost or Out-of-order Bye-bye Messages

In case a device changes its UPnP configuration (adding or removing a service from its description), it announces that it will leave the network by multicasting a bye-bye message, canceling all outstanding advertisements and dropping all current event subscriptions. After changing the configuration, it announces that it rejoins the network (reboot of UPnP device) and sends set of announcements to indicate it is back with a changed configuration.

First of the two problems arise from this approach are shown Figure 5.2. When all bye-bye messages are lost, CPs fail to detect that that a "reboot" has occurred. CPs erroneously assume they are still subscribed to events and that the dynamic state of the device is unchanged. Only when CPs try to renew their event subscription will they recover from this state.

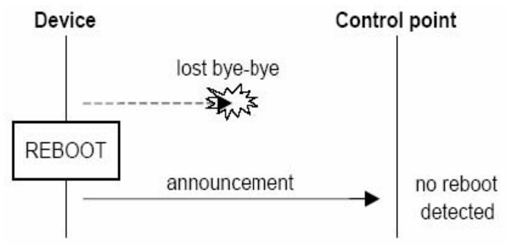


Figure 5.2: Lost Bye-Bye Message Leads to Failure to Detect Reboot

The second problem occurs when bye-bye messages are somehow delayed by the network and when they arrive at out-of-order. Figure 5.3 shows the corresponding message sequence diagram. Since bye-bye message arrives after the new announcement, the CP will incorrectly decide that the device has left the network. Only when the next periodic “alive” message arrives will the CP automatically recover from this state. Although this sequence is unlikely to occur in the relatively small networks UPnP is intended for, IP networks do not guarantee order of arrival of packets.

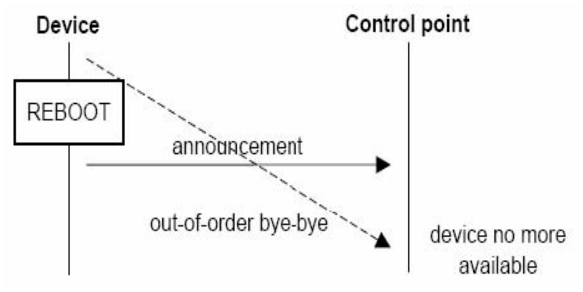


Figure 5.3: Out-of-order Arrival Of Bye-bye Message

UPnP 1.1 discovery contains a new mechanism to enable CPs to detect whether a device has rebooted. Each announcement message includes a bootid.upnp.org header, whose value must increase each time the device first announces on the network, and stays the same in all consecutive announcements. Therefore, each UPnP device must have a persistent state between reboots, like a clock.

Figure 5.4 shows how the "bootid.upnp.org" header solves the "lost bye-bye" problem. A CP remembers the highest bootid.upnp.org value it has received. If it subsequently receives a higher value, it can deduce that a reboot has occurred.

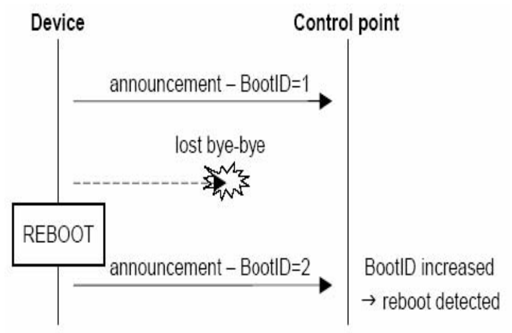


Figure 5.4: Lost Bye-bye Message - Solved

Figure 5.5 shows that the same mechanism enables CPs to discard out-of-order bye-bye messages, since they will contain a lower bootid.upnp.org value than has previously been received.

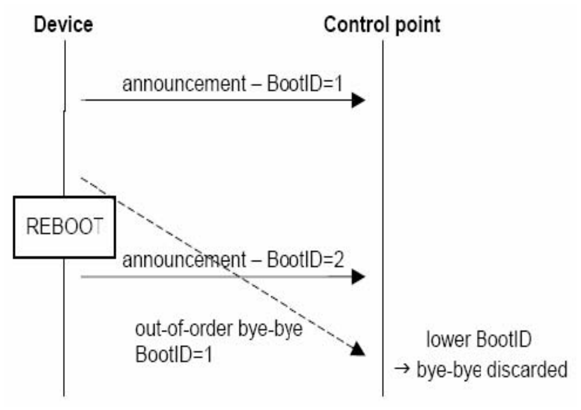


Figure 5.5: Out-of-order Bye-bye- Solved

Finally, as an added bonus the bootid.upnp.org header allows CPs to distinguish between announcements of multi-homed devices (two announcement messages with the same UUID arrive from different IP addresses, but the bootid.upnp.org value is the same) and announcements of devices that “reboot” to change IP address (two announcement messages with the same UUID arrive from different IP addresses, but the bootid.upnp.org value is the different).

5.1.2.4 Persistent Description Caching

If due to a (temporary) disconnection announcements expire, all CPs need to refresh their cache and download the descriptions again. This leads to a peak load on UPnP devices at any reconnection, even if the description information is unchanged. The UPnP 1.1 device architecture adds a configid.upnp.org header to announcements and search responses that improves caching and configid.upnp.org header value characterizes all description files with a single number. Enabling applications to detect whether cached descriptions are still valid: only if the configid.upnp.org value has changed to a new, the CP has to download new description files.

To Avoid race conditions between receiving such announcements and downloading the description files (e.g. suppose that in that period a device changes its configuration), a new “configID” attribute is added to description files. With this attribute a CP can prove that the downloaded description belongs to the current configID. UPnP 1.1 defines a “cached” attribute that can be added to service action descriptions. If the cached attribute of an action is set to 1, the results returned by this action (on invocation) are considered to be a part of the current configuration. As long as the configID is unchanged, these results can be cached and re-used.

5.1.2.5 Unicast M-SEARCH

When bye-bye messages are lost, 30-minute timeouts as fallback is used in UPnP 1.0 SSDP. Many applications require the current view of the network, and want to know within seconds when a specific device disappears from the network. UPnP 1.0 does not offer specific mechanisms for this “liveness” check, and different vendors have different workarounds.

The UPnP 1.1 device architecture introduces unicast searching for fast, efficient device-liveness checks. The format of a unicast M-SEARCH is the same as for the multicast MSEARCH, the only difference is that it is sent to a specific IP address instead of the SSDP multicast address. Few implementations distinguish between multicast and unicast traffic arriving at the same port, and the solution is compatible with most existing UPnP 1.0 implementations.

The response to a unicast search includes bootid.upnp.org and configid.upnp.org headers, providing a quick update of the device state.

During plugfesting of the unicast M-SEARCH, an issue with devices that host multiple UPnP implementations was discovered. So, at most one application can respond to unicast MSEARCH requests on the default port 1900. The searchport.upnp.org header has been defined to enable unicast M-SEARCH requests for multiple independent applications. With this header, applications can announce that they will respond to unicast M-SEARCH requests on a different port.

5.1.3 HTTP Extension Framework and HTTP/1.1 Profile

The use of HTTP extension framework [33] is specified by UPnP 1.0 device architecture. The reasons of deprecation are; the processing model of rfc2774 is intended for one-to-one communication, while discovery is one-to-many, M-POST is rarely used in practice, as it is mandatory to try a normal POST first and rfc2774 never made it through the standards track, but received the “experimental” label, and the WS-I Basic profile forbids use of M-POST [35] and [36].

The UPnP 1.1 device architecture replaces the HTTP extension framework with a simple header-naming scheme that prevents name-clashes by requiring that vendor-defined header names have the following format:

field-name= token “.” domain-name

For example, the UPnP forum uses this format for the new bootid.upnp.org header. This would not clash with a bootid.otherorganization.org header

The UPnP 1.0 device architecture has an unclear HTTP profile that has caused many interoperability problems. The UPnP 1.1 device architecture is explicit in its requirements: a device must be compatible with HTTP/1.0. For backwards compatibility, it must include a host-header even on HTTP/1.0 messages. It is strongly recommended to be compatible with HTTP/1.1 [32].

The UPnP 1.1 HTTP profile explicitly defines the use of the user-agent header and vendor-defined HTTP headers. Using the user-agent header, an UPnP device can identify the highest UPnP version supported by a CP.

5.1.4 Description: Support for Full XML-schema Data Types

In UPnP 1.1, the <URLbase> tag has been quitted, since it’s original definition was unclear and tag had little real usage. How the services used in UPnP 1.0 can be used in a compatible way with UPnP 1.1 or higher, how actions of higher versions of services are to be invoked and the requirement that arguments be sent in the order they are defined, are defined. By this way, unnecessary CP complexity, as CPs have to send the arguments of an action in a different order, depending on the target device is introduced. It typically encodes more complex types in the string-type to deal with limited arguments and results of control actions to a set of simple types (several number-types, a string type and a uuid type). The UPnP 1.1 device architecture adds support for full XML schema defined data types [34], effectively enabling full use of SOAP 1.1.

Additionally, UPnP 1.1 device architecture describes a mechanism that uses the user-agent header in SOAP requests to distinguish between UPnP 1.0 and UPnP 1.1 CPs. This enables services to remain backwards compatible with UPnP 1.0 and still use full XML schema defined data types for new features. The user-agent header is also used to ensure that UPnP 1.0 CPs will not get event notifications of state variables that have an XML schema defined data type.

5.1.5 A SOAP 1.1 Interoperability Profile

Any unknown XML extensions may be skipped during processing in The UPnP 1.0 device architecture conflicting with the SOAP 1.1 processing model: if a header with a “mustUnderstand=1” attribute is unknown, the message body must not be processed. The UPnP 1.1 device architecture uses the SOAP 1.1 processing rules. On the other hand, there is a profile for interoperability. First, incorrectly used “mustUnderstand” attributes (e.g. added to a vendor-defined header) can block interoperability; so headers with a mustUnderstand attribute with value 1 may not be targeted at standardized UPnP services, unless explicitly specified in the standard. Specifically, the mustUnderstand attribute should not be included in messages sent to UPnP 1.0 devices, as they are likely to ignore the attribute. Second, the SOAP “actor” attribute retargets header elements. UPnP headers must be targeted at the endpoint, unless explicitly specified otherwise. Third, vendor-defined header elements within messages that are sent to UPnP 1.0 devices or CPs should not be targeted at intermediaries, since UPnP 1.0 devices and CPs might ignore the actor attribute and parse a header that is not intended for them.

5.1.6 WS-I Basic Profile Compatibility

Compatibility with the WS-I Basic profile would further improve interoperability between devices. The only changes necessary to make the UPnP 1.1 device architecture compatible with the WS-I Basic Profile are;

- The WS-I Basic Profile states that a WSDL 1.1 description must be available for each service [36]. While UPnP 1.1 does not contain a method to retrieve a WSDL document [37], the UPnP description files contain sufficient information to use (e.g.) a style sheet to create a WSDL file from it. In that sense, UPnP 1.1 is compatible with the WS-I Basic Profile.
- The WS-I Basic Profile requires that devices are able to receive both UTF8 and UTF16 encoding. Although, it seemed incompatible with the UPnP 1.0 requirement that senders only use UTF8 the UPnP 1.0 device architecture was silent on capabilities of receivers, despite language to the contrary had been added the UPnP 1.1 draft. By reverting to the UPnP 1.0 requirements, implementations can adhere both to UPnP 1.1 and the WS-I Basic Profile: send only UTF8, but receive both UTF8 and UTF16.
- The WS-I Basic Profile mandates the use of HTTP status code 307 Temporary Redirect, and forbids other redirect methods. The UPnP 1.0 device architecture did not have a clear HTTP profile. The UPnP 1.1 HTTP profile also mandates that 307 Temporary Redirect is implemented, but does not recommend its use.
- The UPnP 1.1 control has explicit SOAP encoding and namespace requirements.

- The WS-I Basic profile forbids the use of the HTTP extension framework. The framework has already been deprecated by UPnP 1.1 as well.
- The UPnP 1.1 SOAP faults were brought in line with the WS-I Basic Profile.
- The WS-I Basic Profile requirement on encoding arrays using XML schema was taken over.

5.2 Web Services

The Internet and Web Technologies have changed the way organizations do business. Nowadays, businesses are willing to put their core business processes on the Internet as a collection of web services [38].

A web service can be defined as “an interface that describes a collection of operations that are network-accessible through standardized XML messaging". They are quickly becoming a significant technology in the evolution of the Web and distributed computing.

Web services can be viewed as a rapidly evolving application-integration technology that allows applications to interact or communicate with each other irrespective of the platform they are running on or the programming language they have been written in. In short, web services are both platform as well as programming language independent. One of the critical factors for the success of web services is that they leverage well-established technologies such as HTTP and XML. They allow developers to create loosely coupled applications based on XML messaging protocols very easily by leveraging the data independence of XML to solve enterprise integration problems, both inside and outside the firewall. Web service interfaces also act like wrappers that can map to any type of software program, middleware system, database management system, or packaged application. They provide a layer of abstraction necessary for bringing together highly distributed and heterogeneous systems.

5.3 Device Profile for Web Services

The Web services architecture includes a rich suite of specifications that provide complementary functions in the areas of security, reliability, and transaction-based messaging. By design, these specifications may be used together to meet varied service requirements. The Web services architecture includes a suite of specifications that define rich functions and that may be composed to meet varied service requirements.

The Devices Profile for Web Services (DPWS) pulls together a core subset of these specifications into an overall whole [39]. The Devices Profile is a detailed specification intended for those who are implementing networking libraries or platform components. It is written as a set of numbered normative statements together with rationale and examples.

The Devices Profile focuses on IP-capable devices which are growing in computational power. Though many of these are still resource-constrained by desktop and server standards, these devices are ready to contribute to general, Web services scenarios involving services already being deployed in the home, small office, enterprise, and Internet. To enable a base level of interoperability between devices and Web services, the Devices Profile calls out best-of-breed Web service specifications in core areas and prescribes how to use them in concert to enable the following networking functions:

- Sending secure messages to and from a Web service
- Dynamically discovering a Web service
- Describing a Web service
- Subscribing to, and receiving events from, a Web service

In each of these areas of scope, this profile defines minimal implementation requirements for compliant Web service implementations.

A very promising approach is that proposed by DPWS, described below. It has the same advantages as UPnP, but additionally it is fully aligned with Web Services technology. DPWS is actually expected to form the foundation for the next major upgrade of UPnP (referred to as UPnP V2), but for reasons of market strategy related to the lack of backwards compatibility, no date is set for this transition. It is further worth noting that Microsoft's next-generation Windows platform ("Longhorn or Vista") will natively integrate DPWS.

The DPWS specification defines an architecture in which devices run two types of services: hosting services and hosted services [40] and [41].

Hosting services are directly associated to a device, and play an important part in the device discovery process. Hosted services are mostly functional and depend on their hosting device for discovery. In addition to these hosted services, DPWS specifies a set of built-in services:

- Discovery services: used by a device connected to a network to advertise itself and to discover other devices.
- Metadata exchange services: provide dynamic access to a device’s hosted services and to their metadata, such as WSDL or XML Schema definitions [37].
- Publish/subscribe eventing services: allowing other devices to subscribe to asynchronous event messages produced by a given service.

In DPWS statement, a controlled device is simply designated as "device" while a controlling device is designated as a "client".

5.3.1 Outline of the DPWS Protocol Stack

The core Web Services standards are the following.

- WSDL (Web Services Description Language) for the abstract description of service interfaces and their binding to transport protocols.
- XML Schema for the definition of the data formats used for constructing the messages addressed to and received from services.
- SOAP, the protocol for transporting service-related messages formatted in accordance with the corresponding WSDL definitions.
- WS-Addressing is closely associated to SOAP and concentrates all message addressing information into the header of the SOAP message envelope, thereby allowing for the message content to be carried over any transport protocol (HTTP, SMTP, TCP, UDP …)
- WS-Policy is used to express policies associated to a Web Service in the form of "policy assertions", complementing the WSDL description of the service.
- WS-MetadataExchange allows for dynamical retrieval of metadata associated to a Web Service (description, schema and policy), thus providing a Web Service introspection mechanism.
- WS-Security is an optional set of mechanisms for ensuring end to-end message integrity, confidentiality and authentication.

The DPWS protocol stack, depicted in Figure 5.6, integrates all the above core standards. With DPWS, all messaging is based on the use of SOAP and WS-Addressing.

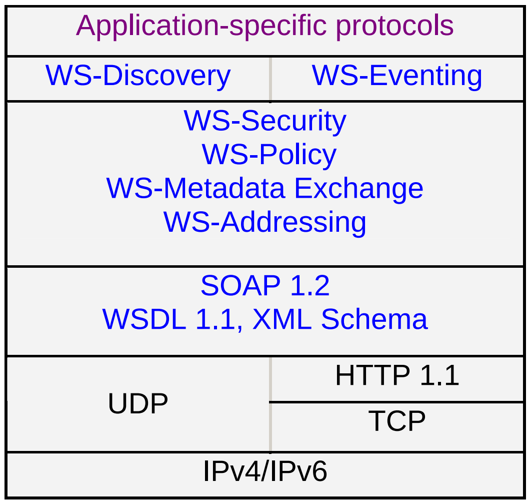


Figure 5.6: Devices Profile for Web Services Protocol Stack

To the above-mentioned Web Services core protocols, DPWS adds Web Services protocols for discovery and eventing.

WS-Discovery is a protocol for plug-and-play, ad hoc discovery of network-connected resources. It defines a multicast protocol to search for and locate devices. Once discovered, a device exposes the services it provides.

WS-Eventing defines a publish-subscribe event handling protocol allowing for one Web Service to subscribe with another Web Service in receiving event notification messages. WS-Eventing is intended to enable implementation of a range of applications, from device-oriented to enterprise-scale publish subscribe systems, on top of the same substrate.

6. UPNP A/V ARCHITECTURE

The UPnP A/V [44] architecture shown in Figure 6.1 defines three main logical entities: a Media Server [45], a Media Renderer [46], and a UPnP A/V CP. The Media Server has access to entertainment content and can send that content to another UPnP A/V device via the network. A Media Renderer is able to receive external content from the network and render it on its local hardware. An A/V CP discovers Media Servers and Media Renderers on the network and using defined UPnP actions, connects a Media Server to the Media Renderer at which point the two devices can stream the content directly from each other without the UPnP A/V CP getting in the way.

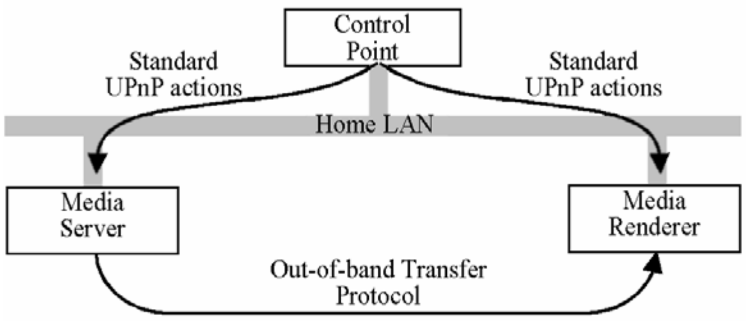


Figure 6.1: UPnP A/V Architecture

The UPnP A/V Architecture was explicitly defined to support the following goals:

- Support arbitrary transfer protocols and content formats
- Direct source-to-sink transfer of content
- Ability to support A/V devices of all complexities

Standard UPnP A/V actions are defined in Media Server and Media Renderer devices making the CP involve only in command and control operations completely independent of the media format or eventual out-of-band transport protocol the devices will use. UPnP A/V also supports scalable devices of all levels of complexity by defining a minimal set of services and actions that each device must support, which ensures basic interoperability and functionality while also defining a set of optional services an actions that would support an extensive set of advanced device capabilities.

As shown in Figure 6.2, the CP is the only component that initiates UPnP actions. The CP requests to configure the Media Server and Media Renderer so that the desired content flows from the Media Server to the Media Renderer (using one of the transfer protocols and data formats that are supported by both the Media Server and Media Renderer). The Media Server and Media Renderer do invoke any UPnP actions to the CP. However, if needed, the Media Server and/or Media Renderer may send event notifications to the CP in order to inform the CP of a change in the Media Server's/Media Renderer's internal state.

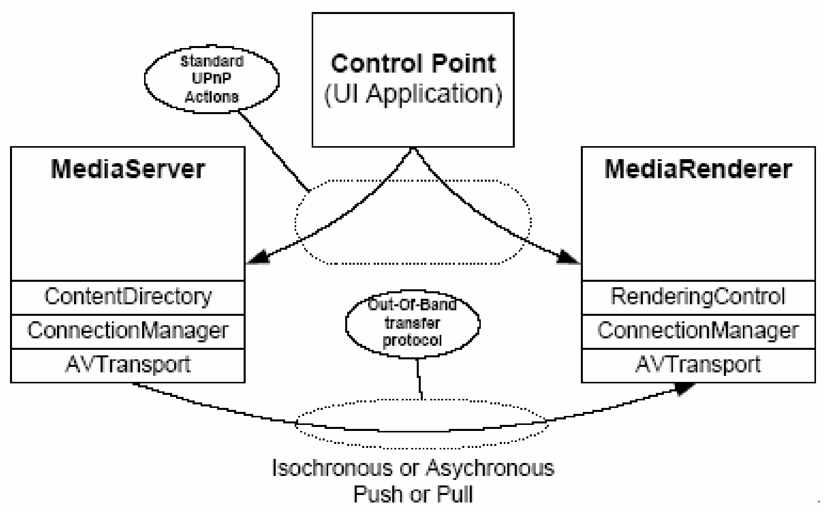


Figure 6.2: Content Playback Scenario

The Media Server and Media Renderer do not control each other via UPnP actions. However, in order to transfer the content, the Media Server and Media Renderer use an "out-of-band" (e.g. a non-UPnP) transfer protocol to directly transmit the content. The CP is not involved in the actual transfer of the content. It simply configures the Media Server and Media Renderer as needed and initiates the transfer of the content. Once the transfer begins, the CP "gets out of the way" and is no longer needed to complete the transfer.

However, if desired by the user, the CP is capable of controlling the flow of the content by invoking various A/VTransport actions such as Stop, Pause, FF, REW, Skip, Scan, etc. Additionally, the CP is also able to control the various rendering characteristics on the Renderer device such as Brightness, Contrast, Volume, Balance, etc.

Although the A/V architecture defines these three logical entities, a physical device may contain any combination of them. For example, many Renderers are likely to include an embedded CP so that the user may control the operation from the same location where the content is rendered.

6.1 Media Server

A Media Server device is used by CPs to search for and locate the content that is available for playback. Media Servers include familiar devices such as VCRs, set-top boxes (cable, satellite, digital broadcast, etc.), camcorders, CD/DVD players/jukeboxes, radio tuners, TV tuners, still-image cameras, etc. A Media Server's primary purpose is to allow CPs to enumerate (i.e. browse or search for) content items that are available for the user to render.

Media Servers contain the Content Directory, Connection Manager, and (optionally) the A/V Transport services (depending on the supported transfer protocols).

6.1.1 Content Directory Service

The Content Directory Service (CDS) [47] allows CPs to discover and enumerate content that is accessible by a Media Server. CDS "content" objects include individual "content items," which represent individual pieces of content such as a song, video clip, or a photo; and "content containers," which represent collections of items such as a play list, CD, or a photo album. Each CDS object, either an item or container, includes meta-data that describe various attributes of the object, such as title, artist, duration, and so forth.

CDS provides both Browse and Search capabilities. CPs that browse a CDS begin at the root of the CDS hierarchy and iteratively examine the structure, container by container, until the desired content item is found. This is similar to how a file system is used to locate a file that is nested several layers down from the root directory. CPs typically use this mechanism when the user does not immediately have a particular content item in mind.

Alternatively, a CP can use the CDS's Search capability to locate all of the items/containers that possess certain attributes (i.e., certain meta-data values such as "creator=Disney").

Part of the meta-data for each object is a list of transfer protocol and data format combinations that are supported for that piece of content. This information is used by the CP in conjunction with the Connection Manager Service on the target Media Renderer to determine which protocols and formats can be used to transfer the content to the Renderer. Each protocol/format combination is identified by a unique Universal Resource Identifier (URI). This URI is used by the CP to identify the content, protocol, and format that are to be used during the transfer.

The data structures defined by CDS dictate the over-the-wire representation of content items/containers and associated meta-data. CDS does not define the Media Server's internal storage mechanisms or structures. Media Servers can store CDS information using any appropriate mechanism. For example, some servers may use a full-featured database system to store its CDS content hierarchy and to provide a rich set of meta-data for each object. Other servers may maintain their CDS information using only a directory/file hierarchy of their internal file system. In this case, the breadth of the meta-data for each object will be limited to the information that is stored by the file system for each directory/file.

As CDS requests are made by the CP, the server converts those requests into a set of operations carried out by the underlying database used to store the CDS hierarchy and meta-data. Depending on the Server's implementation, this may be a set of relational data operations or a set of system calls to the local file system

6.1.2 Connection Manager Service

Connection Manager Service [48] is used to create and manage the sets of Media Renderer connections to the Media Server. The primary action supported by this service is GetProtocolInfo(), allows an A/V CP to retrieve information about what protocols the device supports sending or receiving. The optional PrepareForConnection() action is invoked by the CP to give the Server an opportunity to prepare itself for an upcoming transfer. Depending on the specified transfer protocol and data format, this action may return the InstanceID of an A/VT Service that the CP can use to control the flow of this content (e.g. Stop, Pause, Seek, etc). This InstanceID is used to distinguish multiple (virtual) instances of the A/VT Service, each of which is associated with a particular connection to Renderer. If the PrepareForConnection() action is not implemented, the CP is only able to support a single Renderer at an given time. In this case, the CP should use InstanceID=0.

Multiple (virtual) instances of the A/VT Service allow the Media Server to support multiple renderers at the same time. Devices that do not support PrepareForConnection() by definition support at most one ConnectionID. When the CP wants to terminate this connection, it should invoke the Media Server's ConnectionComplete() action (if implemented) to release the connection.

6.1.3 A/V Transport Service

The optional A/VT Service [49] provides the CP to adjust and control the “playback” of the content stored in the Media Server. Actions in this optional service include standard VCR-like operations of Play, Stop, Pause, Seek, etc. Existence of these actions indicates that Media Server supports the “Pushing” of content to Media Renderer as the control actions are adjusting properties of the stream at its source, not in the renderer.

Although one transport service is advertised, for Media Servers supporting multiple connections, using different InstanceIDs when invoking actions in effect accesses a different logical instance of the advertised A/VT Service for each different InstanceID. A/VT actions take an InstanceID as a parameter that determines to which A/VT Instance the action is applied.

6.2 Media Renderer

A Media Renderer is a device that can receive content from another device and render it using some local hardware. It includes familiar devices such as a TV, a stereo system, a set of speakers, an Electronic Picture Frame (EPF), etc.

Its main feature is that it allows the CP to control how content is rendered (e.g. Brightness, Contrast, Volume, Mute, etc). Additionally, depending on the transfer protocol that is being used to obtain the content from the network, the Media Renderer may also allow the user to control the flow of the content (e.g. Stop, Pause, Seek, etc).

The Media Renderer includes a Rendering Control Service, a ConnectionManager Service, and an optional A/VT Service (depending on which transfer protocols are supported).

6.2.1 Rendering Control Service

Rendering Control Service [50] provides a set of actions that allow the CP to control how the Renderer renders a piece of incoming content. This includes rendering characteristics such as Brightness, Contrast, Volume, Mute, etc. Not all of these actions are required. It doesn’t make sense for an audio renderer to support adjusting the brightness level of a display. These functions are directly related to the capabilities of the output hardware on the Renderer. New instances of the service are created by the ConnectionManager’s PrepareForConnection() action.

The internal logic of RCS is fairly simplistic. As RCS actions are invoked, the RCS simply converts the requested adjustment to the corresponding hardware request as needed.

6.2.2 Connection Manager Service

Connection Manager Service (CMS) is used to manage the connections associated with a device similar to the CMS provided by the Media Server. The primary purpose of the CM service is to allow the CP to query the device, identify and select the common protocol/format that will be used to transfer the desired content from the Server to the Renderer.

The actions defined by the CM service provide a standardized interface to the Server's/Renderer's internal network and media codec subsystems. An implementation of CM must be able to enumerate and configure the transfer protocols that are supported by the device's network subsystem and the data formats that are supported by the device's media codecs according to the requested mechanism. In many cases, this may involve constructing a data path from the device's network subsystem, through the appropriate codec, to the device's output hardware.

A Media Renderer may also implement the optional PrepareForConnection() action. This action is invoked by the CP to give the Renderer an opportunity to prepare itself for an upcoming transfer. Additionally, this action assigns a unique ConnectionID that can be used by a CP to obtain information about the connections that the Media Renderer is using. Also, depending on the specified transfer protocol and data format being used, this action may return a unique A/VT InstanceID that the CP can use to control the flow of the content (e.g. Stop, Pause, Seek, etc). Lastly, PrepareForConnection() also returns a unique Rendering Control InstanceID which can be used by the CP to control the rendering characteristics of the associated content. When the CP wants to terminate a connection, it should invoke the Renderer's ConnectionComplete() action (if implemented) to release the connection.

6.2.3 A/V Transport Service

The A/VT Service provides a number of actions that allow a CP to control the flow of the content. This includes many of the familiar operations typically associated with the mechanical "tape transport" mechanism implemented on most VCRs, such as Play, Stop, Pause, Seek, FF/REW, and so forth.

A/VT also provides the mechanism which the CP uses to identify the content that is to be played. This is done by passing in the unique Universal Resource Identifier (URI), which was obtained from the CDS for the desired content and the selected protocol and format.

Depending on transfer protocols and/or data formats that are supported, the Renderer may or may not implement this service. If the selected protocol is a “pull” model (e.g, HTTP GET), then the renderer is required to provide an instance of A/VT to control the flow of the content (e.g., play, pause, seek). If the selected protocol is a “push” model, then the server must provide an instance of A/VT.

The internal implementation of A/VT Service must hook into the device’s media streaming subsystem in order to configure it to access and stream the desired content and to control that content stream, as directed by the CP (i.e., the end-user). In most cases, the internal logic of the A/VT Service is fairly straightforward. When an A/VT action is invoked, A/VT Service invokes the corresponding operation(s) on the internal media streaming subsystem.

In order to support Media Renderers that can simultaneously handle multiple content items, the A/VT Service may support multiple logical instances of this service. A/VT InstanceIDs are allocated by the ConnectionManager’s PrepareForConnection() action to distinguish between multiple service instances.

6.3 UPnP A/V Control Point

Since UPnP A/V architecture provides no mechanism for UPnP devices to communicate directly with each other, UPnP A/V CP performs all discovery and coordination between Media Servers and Media Renderers, UPnP A/V Architecture defines the external interfaces of the media server and Media Renderer so that a CP can manage the distribution of entertainment content as desired by the end-user. However, A/V Architecture does not define any of the internal structure of the Server/Renderer. This is left entirely to the implementer.

In Figure 6.3, when a CP joins the network, it locates all of the media servers and media Renderers in the network. It does this using SSDP. In order to locate Media Servers in the network, the CP issues an SSDP IP-multicast request packet for any UPnP device that implements the UPnP A/V Media Server device template. All devices that implement the media server template must respond to the request with the URL of its description document. Media Renderers are located in a similar manner.

Once Servers and Renderers are located, the CP obtains and parses each device's XML description document to determine the device's exact capabilities (i.e., its UPnP services, actions, and state variables.) If the device implements the desired capabilities, the CP continues to interact with it.

At some point after the CP initializes itself, a CP may display an initial User Interface (UI) so that the end-user can interact with the CP. The contents and layout of the UI is device-dependent to provide room for innovation and product differentiation.

For each Media Server that is found, the CP uses the Server's CDS to enumerate the content that is available from that Server. CPs often collect CDS information from multiple servers and aggregate it into a single "whole home" view of all of the content that is available from within the home, regardless of which service it is on.

Depending on the CP's UI, the CP will either browse through the CDS information; perform searches on it, or a combination of the above. Once the CP has received and processed the returned data, the CP updates its UI.

After the user has selected the desired content, the CP determines which transfer protocols and data formats are supported for that particular piece of content. This is done by examining the CDS meta-data for the selected item. Using the Connection Manager Service on each Renderer, the CP can obtain the set of protocols/formats that are supported. The CP then compares the protocol/format information from the server's CDS and the Renderer's CM to determine which Renderer(s) is capable of rendering the desired content.

After a common protocol/format has been identified, the CP invokes the CMSs on both the Server and Renderer to inform each device of the target protocol/format. In response, the CMS would set up and configure its internal network and media streaming subsystems based on the common protocol/format that has been chosen.

As a result of configuring each device, either the Server or Renderer will return an instance of the A/VT Service that is associated with the media stream that has just been set up. The CP uses the returned A/VT to specify the content that is to be transferred from the Server to the Renderer.

When the user indicates the desired operation that is to be performed on the current content (e.g., Play, Seek, etc.), the A/VT Service is invoked accordingly. After the content has begun to play, the user may select other operations (Stop, Pause, Seek, etc.), any time during the streaming session.

As the content is being rendered, the CP may provide a set of UI components that allow the user to control how the content is rendered. This includes various rendering characteristics such as loudness of the volume, brightness of the video/image, etc. As the user adjusts various rendering characteristics, the CP invokes the appropriate action on the RCS.

The core capabilities needed to distribute and render the selected content on a particular rendering device is provided by the UPnP A/V Architecture. The CP developer can concentrate on providing innovative and compelling media distribution UIs for the end-user.

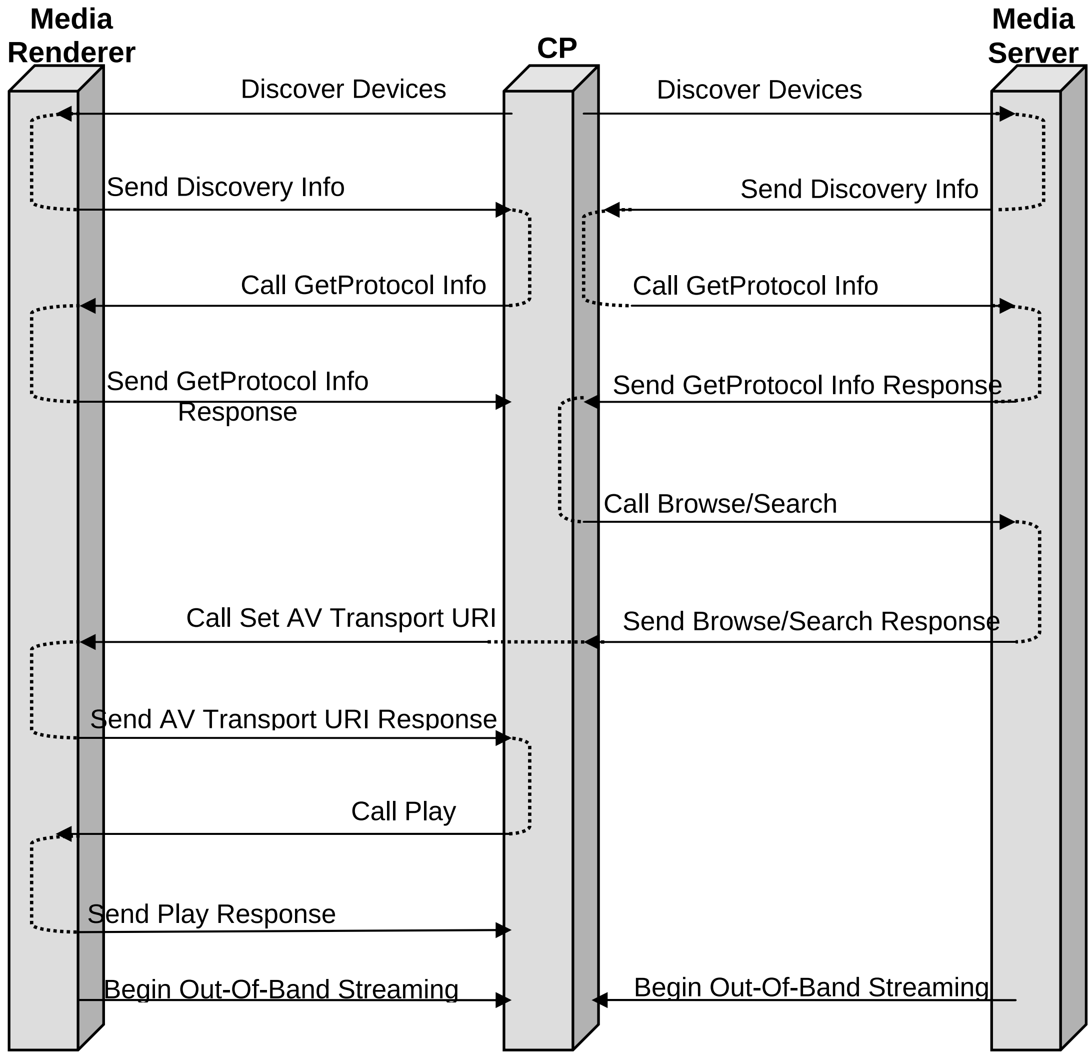

Media Renderer
CP
Media Server
Discover Devices
Discover Devices
Send Discovery Info
Send Discovery Info
Call GetProtocol Info
Call GetProtocol Info
Send GetProtocol Info Response
Send GetProtocol Info Response
Call Browse/Search
Call Set AV Transport URI
Send Browse/Search Response
Send AV Transport URI Response
Call Play
Send Play Response
Begin Out-Of-Band Streaming
Begin Out-Of-Band Streaming

Figure 6.3: Interaction between Devices and CP

7. IMPLEMENTATION

The implementation is based on making a device UPnP enabled that is originally a non-UPnP device. An ordinary television has been made UPnP technology enabled device as a Media Renderer. Since UPnP technology is independent of physical layer and OS, a wireless technology (802.11 b/g) has been chosen for physical communication of Devices and CP, and Linux OS for Media Renderer's hardware interface and Windows XP OS for Media Server and CP applications. Although the A/V Architecture defines these three logical entities, a physical device may contain any combination of them. For that project, laptop has been designed as a physical device that contains CP and Media Server applications. Media Renderer hardware interface has been designed and installed Linux OS (Fedora Core6) and is connected to Rendering Device which is an ordinary TV. By doing that, UPnP A/V Architecture is implemented in accordance with the overall UPnP technology. The layout of the implementation, user interfaces of CP and devices and presentation pages devices are as shown in Figure 7.1.

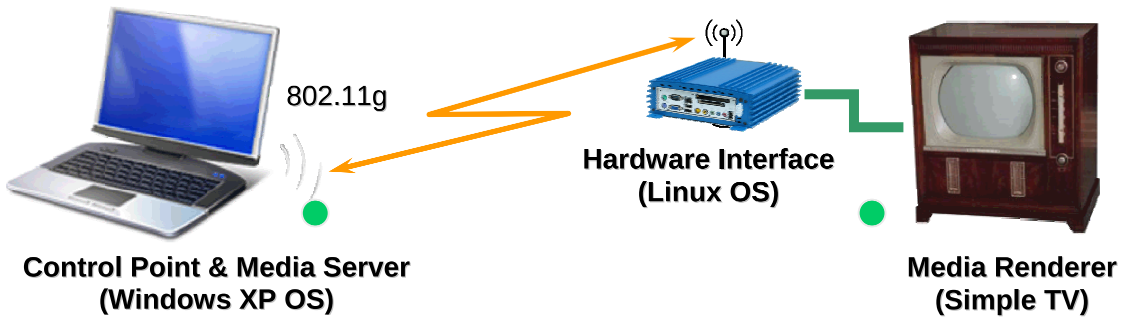


Figure 7.1.a: Layout of the Implementation

The addressing protocol built in to UPnP devices allows them to join an IP network dynamically by acquiring an address without user configuration either in managed or ad-hoc networks. In unmanaged or ad-hoc network, network nodes themselves make up the network. In managed network, network allows devices to acquire an IP address from a DHCP server on the network. In implementation phase a wireless ADSL Router (A DHCP Server as well) is used for devices to acquire IP addresses. When the CP and

Media Renderer Device join the network, acquire a unique address. When a device (Media Server or Renderer) comes online, announces its presence. This lets CP know that the device(s) is available. If a CP comes online after devices has announced their presence, sends out discovery requests. The CP does not need to repeat the discovery request because any device that come online after it has issued the request will announce their presence. When CP is connected the network after the presence announcements of Media Server and Renderer, it locates Media Server and Media Renderer in the network. It does this using SSDP. In order to locate Media Server in the network, CP issues an SSDP IP-multicast request packet for the UPnP device that implements the UPnP A/V Media Server device template with a Media Server NT Header Type. The device that implements the Media Server template must respond to the request with the URL of its device description document. The device description document contains device information such as manufacturer, make, model, and serial number; a list of services provided by the device; and a list of embedded devices. Media Renderer is located in a similar manner. Once Server and Renderer are located, CP obtains and parses each device's XML device description document and gets the exact URL addresses of service description document of both Media Server and Renderer to determine the their exact capabilities (i.e., its UPnP services, actions, and state variables.). *A* service description document contains detailed information about a device's service, the actions that service provides, and the service's parameters and return value. After the CP initializes itself, it may display an initial UI, so that the end-user can interact with the CP. The contents and layout of the UI is device-dependent to provide room for innovation and product differentiation. An UI that belongs to CP was prepared for discovery of Media Server and Media Renderer that have already advertised them to the network. In implementation example, discovery of Media Server and Renderer is fulfilled by pressing the "Start Server Discovery" button and "Start Renderer Discovery" button on the CP User interface as shown in figure 7.1.b. After getting responses from related devices, they are listed in "List of Video Servers" and "List of Video Renderers" list boxes.

Once a CP discovers devices and retrieves their description document, it may want to control one or more of the services contained in the device. The CP creates the XML document and posts it to the control URL for the service, as specified in the description document. The CP can request current values and make changes to the service's state table. On the server side, the control server waits for control requests. The control server is an HTTP -like server implementing the SOAP protocol. A device can operate more than one control server depending on the combination of services provided by the device. For Media Server that is found, depending on the CP's UI, the CP will either browse the video file information; perform searches on it, or a combination of the above. Once the CP has received and processed the returned data, CP updates its UI. As for implementation phase, “Play”, “Pause”, “Stop” and “Set Volume” actions for Media Renderer and “Browse”, “Search”, ”Start Stream” and “Stop Stream” actions for Media Renderer have been used for Controlling Media Server and Renderer via CP user Interface. After the user has selected the desired content that is any type of video file for testing, CP determines which transfer protocols and data formats are supported for that particular piece of content. After a common protocol/format has been identified, CP invokes the services on both the Server and Renderer to inform each device of the target protocol/format. In response, the services on both Media Server and Renderer would set up and configure its internal network and media streaming subsystems based on the common protocol/format that has been chosen. VLC Player has been chosen for internal media streaming subsystem of the network. When the user indicates the desired operation that is to be performed on the current content (e.g., Play, Seek, etc.), the A/VT Service is invoked accordingly. After the content has begun to play on TV screen, the user may select other operations (Stop, Pause, Seek, etc.), any time during the streaming session. As the content is being rendered, CP may provide a set of UI components that allow the user to control how the content is rendered. This includes various rendering characteristics such as loudness of the volume, brightness of the video/image, etc. As the user adjusts various rendering characteristics, the CP invokes the appropriate action on the RCS.

If the services provided by Media Server and Renderer have “evented state variables”, UPnP Eventing will exist. CP sends subscription messages to Media Server and/or Renderer which are called as publishers. If publishers accept the subscription then send subscription accepted message with Subscription ID and duration first and send initial event message to subscriber including names and values of all evented variables provided by the service as a convenience to allow CP to initialize its representation of device's state. To keep a subscription active, CP sends a renewal subscription message before that subscription expires. If the device implements the desired capabilities, the CP continues to interact with it. As a default, subscription renewal period is decided to be 90 seconds.

When a subscription expires, the subscription identifier becomes invalid, and the publisher stops sending event messages to that subscriber. If the subscriber tries to send any message other than a subscription request message - a subscription renewal request or a subscription cancellation message - the publisher rejects that message due to the invalid subscription identifier.

When eventing is no longer needed from a particular service, a cancellation message should be sent to that service's eventing URL. If CP is removed abruptly from the network, it might be impossible to send a cancellation message. As a fallback, the subscription will eventually expire on its own unless renewed. To cancel a subscription to eventing for a service, a subscriber should send a request with method UNSUBSCRIBE as described in eventing phase.

When an evented state variable changes, the service notifies subscribed CPs by sending event messages. Event messages are sent as soon as possible after the state changes so that CPs receive accurate and timely information about the service, allowing them to display a responsive user interface. In order to prove that, two or more CP is initiated and one of them is checked as “Do not renew subscription” in CP user interface. In 90 seconds, changing the value of the volume is evented to all CPs with the new value, but after 90 seconds, the “Do not renew subscription” checked CP is not evented about the new value of the volume.

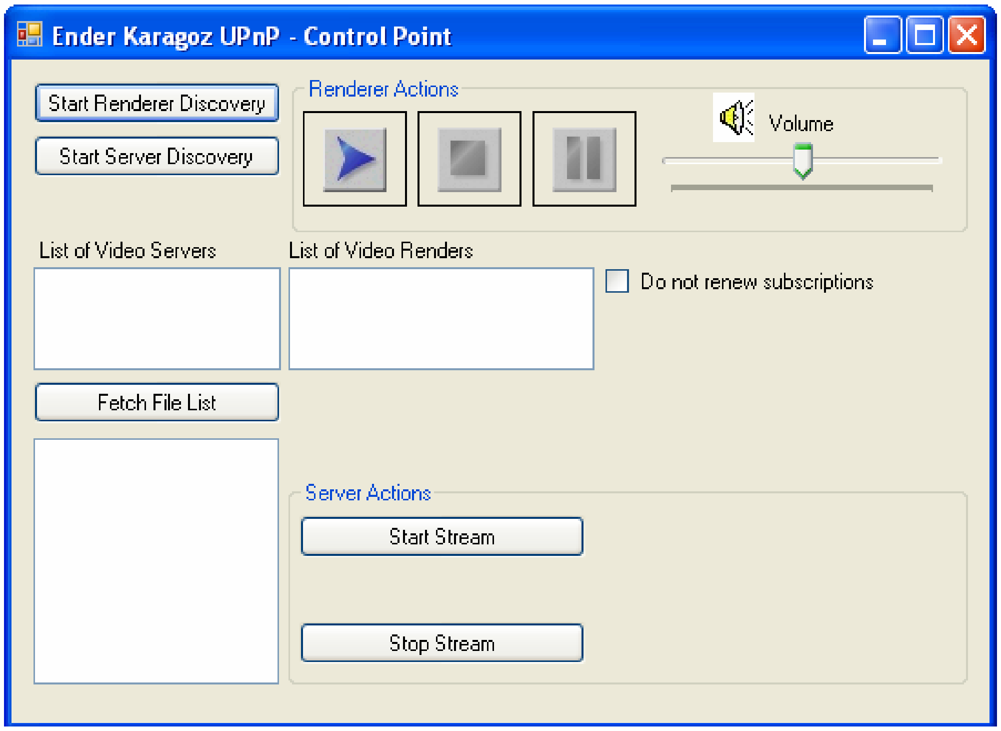
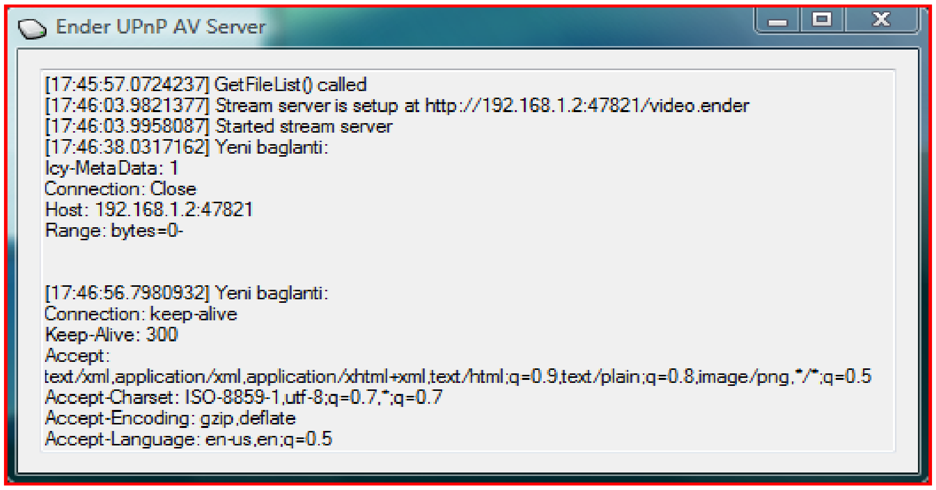


Figure 7.1.b: Control Point User Interface


Ender UPnP AV Server

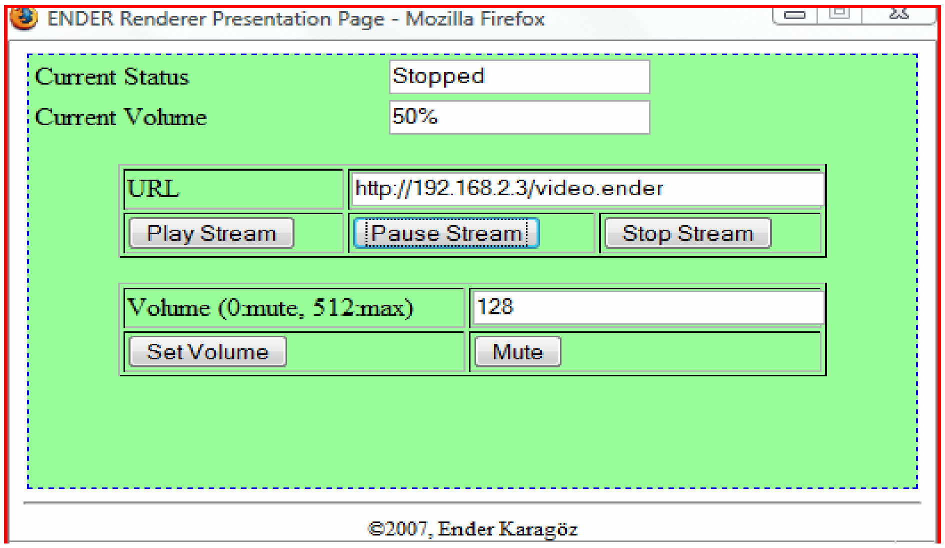
```
[17:45:57.0724237] GetFileList() called
[17:46:03.9821377] Stream server is setup at http://192.168.1.2:47821/video.ender
[17:46:03.9958087] Started stream server
[17:46:38.0317162] Yeni baglanti:
Icy-MetaData: 1
Connection: Close
Host: 192.168.1.2:47821
Range: bytes=0-


[17:46:56.7980932] Yeni baglanti:
Connection: keep-alive
Keep-Alive: 300
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Accept-Encoding: gzip,deflate
Accept-Language: en-us,en;q=0.5
```

Figure 7.1.c: Media Server User interface

Devices have been enabled to use their internal web servers to provide a web interface for management and control of the devices. So, presentation pages for Media Server and Renderer has been also designed in order to control and get last updates of these devices.

A device presents a browser-based user interface for manual user control to allow viewing of that device's status, service information or to change operational parameters by sending SOAP requests to devices. Each UPnP device contains an internal HTTP Web server. Presentation Page of Media Renderer is as shown in Figure 7.1.d. and Media Server in Figure 7.1.e.

In implementation phase, a Media Renderer and a Media Server presentation pages are used for controlling the actions on these devices and getting last updates about evented state variables. It is optional that, all A/V architecture may be implemented via these presentation pages or they may be passive interfaces just for getting the last updates of these devices. In the implementation example, all actions can be invoked and received last updates by using “current Status” and “Current Volume” textboxes via Media Renderer and Server presentation page interfaces.


ENDER Renderer Presentation Page - Mozilla Firefox

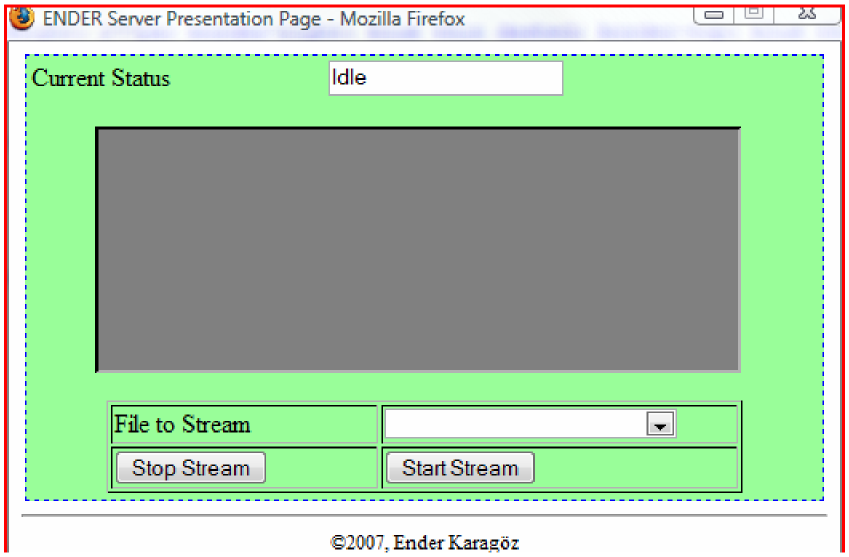
Current Status	Stopped
Current Volume	50%

URL	http://192.168.2.3/video.ender	
Play Stream	Pause Stream	Stop Stream

Volume (0:mute, 512:max)	128
Set Volume	Mute

©2007, Ender Karagöz

Figure 7.1.d: Media Renderer Presentation Page


ENDER Server Presentation Page - Mozilla Firefox

Current Status	Idle

File to Stream	
Stop Stream	Start Stream

©2007, Ender Karagöz

Figure 7.1.e: Media Server Presentation Page

Since the asynchronous nature of network device interaction inherently makes it difficult to debug the code in the synchronous step-by-step fashion of user application development, it is difficult to develop UPnP devices and figure out exactly why UPnP device is not working when one cannot get visibility into what is happening over the network. Intel has released a set of tools that are useful during UPnP device development which assist hardware and software developers in accelerating their development testing and deployment of devices that comply with UPnP standards [51]. Built on MS .NET technologies, the toolkit provides a wide set of useful applications

(Device Author, Spy, Sniffer, Validator, Relay etc.) that are invaluable for UPnP device development.

Besides using C# Programming Language and Visual Studio.Net Platform, these tools are used as a reference during the implementation phases of writing the code of CP and devices. The overall implementation code is burned into a CD which is an annex of this thesis.

8. CONCLUSION AND THE FUTUREWORK

HAN is the collection of elements that process, manage, transport, and store information, enabling the interconnection, interoperation and integration of multiple computing, control, monitoring devices such as home electronic appliances, entertainment devices, PC hardware, and telecommunication devices and involves distribution of audio, video, and data around the home. It allows shared resources, and enables information to be sent between devices and the narrowband or broadband access into the home to be shared.

Universal Plug and Play offers advantages for home networking applications, including leveraging existing Internet standards that are already widely deployed with proven benefits and universality that allows many different physical networks to connect various types of devices using any platform and any operating system.

Future directions for this technology might entail security enhancements, coexistence with other standards, and additional or expanded DCPs. Although the UPnP version1 DCPs address many device classes, numerous other devices types could be candidates for future DCPs. The UPnP Forum actively solicits new members with expertise in devices that could benefit from UPnP technology and welcomes proposals for new DCPs.

The open UPnP architecture is supported by many companies from multiple industries. These companies are developing UPnP products and participating in the definition and refinement of the technology. The architecture builds on existing standards and emphasizes ease of use. As such, UPnP technology is important for home networking applications.

Security is no longer an option and could even be used to promote advanced features. UPnP is based on IP-based technologies that are currently used by pirates to gain access to valuable equipments inside the end-users' houses. The security of the whole system, from the service provider to the end user’s device is considered. UPnP protocol is a good solution to be used inside the house to provide a secure communication mean with the devices. It is of the utter importance to start to develop and provide UPnP devices that embed security (before it is too widely deployed).

UPnP 1.1 device architecture is compatible with the WS-I Basic Profile, a next step can be to create a new version of the architecture that requires improved performance and reliability, elimination of ambiguities and better interoperability, and compatibility with external standards to simplify implementation efforts and to simplify adoption of new technologies in the device architecture. Next step to grow the UPnP 1.1 device architecture is to investigate which other Web Services components can add value to the UPnP device architecture. To qualify for use in the UPnP device architecture, components should not be drafts, but stable and agreed-on standards

The UPnP 1.1 device architecture does not include a discovery server yet. By defining such a discovery server as a normal UPnP device class, bootstrapping (discovering the discovery server) can use SSDP, and control and eventing mechanisms are also available. As a normal device class, this would not even require an update of the basic architecture.

The UPnP 1.1 eventing specification is not based on SOAP, but does use XML. The publish/subscribe patterns are implemented using normal actions could be done in UPnP: each UPnP service can define a “subscribe” and “unsubscribe” action, with as one input-parameter a SOAP.

A very promising approach is that proposed DPWS. It has the same advantages as UPnP, but additionally it is fully aligned with Web Services technology. DPWS is actually expected to form the foundation for the next major upgrade of UPnP (referred to as UPnP V2), but market strategy related to the lack of backwards compatibility is the problem of that technology. It is further worth noting that Microsoft's next-generation Windows platform ("Longhorn or Vista") natively integrates DPWS.

UPnP A/V Architecture defines the foundational interfaces that enable device manufacturers to develop and deploy multimedia products that interoperate with devices from other manufactures. Since A/V Architecture already defined the fundamental interoperability mechanisms, manufacturers are free to focus their energies on building innovative capabilities into their products. This helps simplify the development process and allow vendors to provide self-configuring, interoperable products to the marketplace with lower development costs. The combination of lower development and installations costs enables products to be brought to the marketplace at pricepoints and convenience levels that allow for mass market adoption of these products. Consequently, UPnP A/V Architecture makes it possible for mass market consumers to finally realize the compelling benefits of being able to access rich multimedia content “anytime, anywhere”.

REFERENCES

[1] **Turnbull, J. G.**, 2002. Introducing Home Area Networks, BT Technology Journal, **20**, 2, 30-38

[2] **Teger, S and Waks, D.J.**, 2002. End-User Perspectives on Home Networking, IEEE Communications Magazine, 114-119.

[3] **Rose, B.,** 2001. Home Networks: A Standart Perspective, IEEE Communication Magazine, 78-85.

[4] **Adams, C.E.**,2002. Home Area Network Technologies, BT Technology Journal, **20**, 2, 53-72.

[5] **George, A.**, 1999.Residential Broadband, pp.284-318, Cisco Press, Indianapolis.

[6] **Vaxevanakis, K.**, **Zahariadis**, **Th. and Vogiatzis**, **N**, 2002. A Review on Wireless Home Network Technologies,Mobile Computing and Communications Review, **7**, 2, 59-68.

[7] **Gast, M.**, 2005. 802.11 Wireless Networks The Definitive Guide, O'Reilly Media, California.

[8] **Ganz, A.**, **Ganz, Z.**, **and Wongthavarawat, K.**, 2003. Multimedia Wireless Networks: Technologies, Standards, and QoS, New Jersey.

[9] **Chen, W. Y.**, 2003. Home Networking basis: Transmission Environments and Wired/Wireless Protocols, Prentice Hall, New Jersey.

[10] Ultra-Wideband (UWB Technology) Enabling high-speed wireless personal area networks, http://www.intel.com/technology/ultrawideband/downloads/Ultra-Wideband.pdf

[11] ZigBee Alliance, http://www.zigbee.com/

[12] Bluetooth Core Specification, http://www.bluetooth.com

[13] Universal Plug and Play Forum, http://www.upnp.org/

[14] Universal Plug and Play Implementers Corporation, http://www.upnp-ic.org

[15] **Jeronimo, M. and Weast, J.**, 2003. Universal Plug and Play: The Foundation of the Digital Home Technology, Technology@Intel Magazine.

[16] **Miller, B. A.**, **Nixon, T.**, **Tai, C. and Wood, M.D.**, 2001. Home Networking with Universal Plug and Play, IEEE Communication Magazine.

[17] **Jeronimo, M. and Weast, J.**, 2003. UPnP Design by Example: A Software Developer's Guide to Universal Plug and Play, Intel Press, Oregon.

[18] Extensible Markup Language, http://www.w3.org/XML/

[19] Simple Object Access Protocol, http://www.w3.org/TR/2000/NOTE-SOAP-20000508.

[20] Universal Plug and Play Device Architecture, http://www.upnp.org/specs/arch/UPnP-DeviceArchitecture-v1.0-20060720.pdf.

[21] Home Audio Video Interoperability, http://www.havi.org

[22] Open Services Gateway Initiative, http://www.osgi.org

[23] The LonWorks Platform, http://www.echelon.com/developers/lonworks/default.htm

[24] Simple UPnP Proxy Protocol Architecture, http://www.altotec.de/SUPP/SUPPA10_20020204.pdf

[25] Microsoft Corporation, 2005. An Overview of the Simple Control Protocol (SCP), http://www.microsoft.com/japan/windows/scp/default.mspx

[26] **Ellison, C.**, 2002. Home Network Security, Intel Technology Journal **6**, 4, 37-48.

[27] DeviceSecurity: 1 Service Template http://www.upnp.org/standardizeddcps/documents/DeviceSecurity_1.0cc_001.pdf

[28] Security Console :1 Service Template http://www.upnp.org/standardizeddcps/documents/SecurityConsole_1.0cc.pdf

[29] UPnP Security Ceremonies Design Document http://www.upnp.org/download/standardizeddcps/UPnPSecurityCeremonies_1_0secure.pdf

[30] **Maarten, P. B.**, 2005. UPnP 1.1 – Designing for Performance & Compatibility IEEE Transactions on Consumer Electronics, **51-1**,69-75.

[31] **Gokulrangan, V.R.**, 2002. Internetworking Using IPv6 Technology Inside and Outside the Home, Intel Technology Journal, **6**, 4, 69-77.

[32] HTTP: Hypertext Transfer Protocol 1.1, http://www.ietf.org/rfc/rfc2616.txt.

[33] HTTP extension framework, http://www.ietf.org/rfc/rfc2774.txt.

[34] XML Schema Part 2: Datatypes, http://www.w3.org/TR/xmlschema-2/

[35] Deprecating Site Local Addresses, http://www.ietf.org/rfc/rfc3879.txt

[36] WS-I Basic Profile http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04- 16.html

[37] **Newcomer, E.**, 2002. Understanding Web Services, Addison-Wesley Press, Boston.

[38] Web Services, http://www.w3.org/2002/ws/

[39] Microsoft Corporation, 2006. Devices Profile for Web Services. http://specs.xmlsoap.org/ws/2006/02/devprof/devicesprofile.pdf

[40] A Technical Introduction to the Devices Profile for Web Services http://msdn2.microsoft.com/en-us/library/ms996400.aspx

[41] **Smit, H.**, **Mensch, A. and Jammes, F.**, 2005. Service-Oriented Device CommunicationsUsing the Devices Profile for Web Services, The Pervasive and Global Computing Group, November 28 – December 2, Grenoble, France.

[42]. **Jongwoo, S.**, **Daeyoung, K.**, **Hyungjoo, S.**, **Junghyun, K.**, **Seong, Y. L. and Choi, J. S.**, 2006. UPnP Based Intelligent Multimedia Service Architecture for Digital Home Network, Proceedings of the Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems and Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06).

[43] **Rasheed, Y. and Ritchie, J.**,2002. High-Quality Media Distribution in the Digital Home, Intel Technology Journal, Vol. 6, Issue 4.

[44] UPnP A/V Architecture: 1 http://www.upnp.org/specs/av/UPnP-av-AVArchitecture-v1-20020622.pdf

[45] MediaServer:2 Device Template Version 1.01 http://www.upnp.org/specs/av/UPnP-av-MediaServer-v2-Device-20060531.pdf

[46] MediaRenderer:2 Device Template Version 1.01
http://www.upnp.org/specs/av/UPnP-av-MediaRenderer-v2-Device
20060531.pdf

[47] ContentDirectory:2 Service Template Version1.01
http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v2-Service-
20060531.pdf

[48] ConnectionManager:2 Service Template Version
http://www.upnp.org/specs/av/UPnP-av-ConnectionManager-v2-Service-
20060531.pdf

[49] AVTransport:2 Service Template Version 1.0.1
http://www.upnp.org/specs/av/UPnP-av-AVTransport-v2-Service-20060531.pdf

[50] RenderingControl:2 Service Template Version 1.0.1
http://www.upnp.org/specs/av/UPnP-av-RenderingControl-v2-Service-
20060531.pdf

[51] Intel® Authoring Tools for UPnP Technologies
http://www.intel.com/cd/ids/developer/asmona/eng/downloads/upnp/overview/in
dex.htm

BIOGRAPHY

Nuri Ender KARAGÖZ was born in Balıkesir, on the 24 th of April, 1970. He received his B.Sc. degree from War Academy, Electric and Electronic Department in 1992. Upon graduation, he had a special education programme on computer science for a year in Computer Engineering Department of Middle East Technical University in 1998. He worked as a chief system administrator of 15th Corps Command Headquarters in Kocaeli for four years on developing Web based, database applications and military automation projects and project management issues. He was appointed to 3rd Corps Headquarters Command as Information System Manager Officer in 2003. He worked in Excercise and Peace Headquarters Information systems, installation and system management of mobile local and wide area networks, Web based and database applications and management of military automation projects.

He has been married since July 2000 and has one two-and-half years old boy.