The NEMO P2P Service Orchestration Framework

William B. Bradley, David P. Maher
InterTrust Technologies
{wbradley, dpm}@intertrust.com

Abstract

NEMO (Networked Environment for Media Orchestration) is a policy managed, peer-to-peer, service orchestration framework that supports the formation of grass roots, self organizing service networks that can enable rich media experiences. We describe some of the interesting features of the NEMO architecture, and some experimental applications. Services are distributed across peer-to-peer communicating nodes. Each node provides message routing and orchestration using a message pump and workflow collator designed specifically for NEMO. Distributed policy management of service interfaces helps to provide trust and security, supporting commercial exchange of value.

P2P messaging and workflow collation allow rich services to be dynamically created from a heterogeneous set of primitive services. We examine the possibilities of P2P computing when the shared resources are services of many different types, using different service interface bindings beyond those typically supported in a web service deployment built on UDDI, SOAP, and WSDL. NEMO provides a media services framework enabling nodes to find one other, interact, exchange value, and cooperate across tiers of networks from WANs to PANs. Using NEMO, digital rights management is more concerned with policy management of service interfaces than copy protection, and new services can more easily ensure fair use. This has the potential to change the tension between consumers and content providers in the digital content domain as the NEMO enriched infrastructure provides consumers with better information, more useful services and instant gratification.

1. Introduction

NEMO is an experimental services framework that enables various stakeholders in the consumer or enterprise media space (consumers, content providers, device manufacturers, service providers, enterprise departments) to find one another, establish a trusted relationship, and exchange value in rich and dynamic ways through trusted service interfaces. It is a platform for interoperable, secure, media related ecommerce in a heterogeneous world of consumer devices, media formats, communication protocols and security mechanisms. While many people have articulated a goal for media distribution where any content is available to anyone, anytime, anywhere on any useful device using viable business models, significant barriers exist to the goal of an interoperable and secure world of media related services:

- Overlapping defacto and formal standards that inhibit interoperability.
- Web services standards and methods that do not bridge services spanning web distribution and personal area network protocols
- Consumer devices that cannot locate and connect to needed services
- Non interoperable implementation technologies
- Impedance mismatches between different trust and protection models
- No unified notion of content governance useful in a P2P distribution model

While emerging web service standards are beginning to address these issues on the web, our goal is to address them across multiple tiers of network nodes spanning personal and local area networks, home, enterprise, and department gateways, and wide area networks. Furthermore, we are interested in the dynamic, peer-to-peer orchestration of both simple and complex services using a variety of local and remote interface bindings (e.g. WS-I [1], Java RMI, DCOM, C, .Net, etc.) allowing us to integrate legacy applications. We also want to be able to use Bluetooth [15], UPnP [5], Rendezvous [4], JINI [12], UDDI [11], etc. all in the same service where each node uses the discovery service(s) most appropriate for the device that hosts the node.

In the media world, the systems and interfaces required or favored by the stakeholders (publishers, distributors, service providers, device providers, and consumers) differ widely. Ideally, our approach will unite the capabilities provided by these entities into

integrated services that can rapidly evolve into optimal configurations meeting the needs of all participating entities. This approach is based on P2P service orchestration.

While we have seen the advantages of P2P frameworks for such things as music and video distribution, we believe that we can use P2P technology much more extensively. Most activity in web services has focused on machine-to-machine interaction with relatively static network configuration and client service interactions. We focus on situations where a person carries parts of her personal area network, moves into the proximity of a LAN or another PAN, and wants to immediately reconfigure service access, and connect to many additional services on a peer basis. Thus, we see the need to integrate web services more coherently with technologies from network services and consumer electronics. We also see opportunities in media enterprise services. While enterprises are most often organized hierarchically, and their information systems can reflect that organization, when people in two enterprises interact, they often will do so more effectively through peer interfaces. However, so far, we have not seen peer-to-peer frameworks allow one enterprise to securely expose its various service interfaces to its customers and suppliers in such a way as to allow those entities to interact at natural peering levels, allowing those entities to orchestrate the enterprise's services in ways that best suit them. This requires trust management of those peer interfaces. The approach that we describe in this paper will allow this.

2. Concepts and Definitions

Service – Any well defined functionality offered by some provider. This could be a low level service offered within a device such as a cell phone (e.g. a voice recognition service), or a multi-faceted service offered over the worldwide web (e.g. a shopping service). Some types of anticipated service providers include consumer electronic devices such as cell phones, PDAs, portable media players, and home gateways; network operators such as cable head-ends and cellular network providers; retailers such web-based stores and content license providers.

Service Interface Binding – The conventions and protocols used to invoke a service. These may be represented in a variety of ways, such as WS-I XML, RPC based on the WSDL [2] definition or a function invocation from a DLL.

Service Orchestration – The assembly and coordination of services into manageable coarser grained services, reusable components, or applications guaranteed to adhere to rules specified by the service provider.

Governance – The process of exercising authoritative or dominating influence over some item such as a music file, a document, or a service operation such as the ability to download and install a software upgrade. Governance typically interacts with trust management, policy management, and content protection.

Peer-to-Peer (P2P) Philosophy – A communication model supporting symmetric access to services for all participants.

NEMO Node – A participant within the NEMO Framework. A node may act in multiple roles as a service consumer and/or a service provider. Nodes may be implemented in a variety of forms including consumer electronic devices, software agents such as media players, or virtual service providers such as content search engines, DRM license providers, or content lockers. NEMO services may be orchestrated to provide more robust composite services.

NEMO interface – an interface, defined using WSDL, that is part of or extends any of the interfaces defined in the NEMO service profiles

Service Access Point (SAP) – encapsulates the functionality necessary for allowing a NEMO node to make a service invocation request to a targeted set of service-providing NEMO nodes and receive a set of responses.

Workflow Collator (WFC) – provides a common interface allowing a node to manage and process collections of request and responses. This interface provides the basic building blocks to orchestrate services through management of the messages associated with the services.

NEMO Trust Management (TM) – provides a common set of conventions and protocols for creating authorized and trusted contexts for interactions between NEMO nodes (peers). NEMO TM may leverage and/or extend other existing security specifications and mechanisms including WS-Security [13] and WS-Policy in the web services domain.

3. Logical Model

An instance of the NEMO framework consists of a logically connected set of nodes that interact in a P2P fashion. NEMO nodes interact by making service invocation requests and receiving responses. The format and payload of the request and response messages is defined in XML. The NEMO Framework supports the construction of diverse communication patterns ranging from direct interaction with a single service provider to a complex aggregation of a choreographed set of services from multiple service

providers. The Framework supports the basic mechanisms for using existing service choreography standards and allows service providers to use their own conventions.

A service interface may have one or more service bindings. A NEMO node may invoke the interface of another node as long as that node's interface binding is described and the requesting node can support the conventions and protocols associated with the binding. E.g., if a node supports a web service interface, a requesting node may be required to support SOAP, HTTP, WS-Security, etc. Any service interface may be controlled in a standardized fashion directly providing aspects of rights management. All interactions between NEMO nodes can be viewed as governed operations.

Any type of device (physical or virtual) as potentially may be NEMO-enabled. At a coarse level of granularity, a NEMO-enabled device exposes a NEMO services API and the corresponding set of service implementations. It may also include a NEMO service adaptation layer exposing a subset of an entity's native services accessed using one or more discoverable bindings. This provides a level of abstraction above the native services API so that a service provider can more easily support multiple types of service interface bindings. In situations where a service adaptation layer is not present, it may still be possible to interact with the service directly through a service access point if it supports the necessary communication protocols. A node may also include modules exported from the NEMO Framework SDK – which contains components that provide functionality for working with the NEMO framework including a Service Access Point and Workflow Collator.

The actual design and implementation corresponding to each of the above modules will vary from device to device.

4. NEMO Service Description

The NEMO architecture requires a flexible and portable way of describing the syntax of requests and responses associated with service invocation, data types used within the framework, message enveloping, and data values exposed by and used within the NEMO framework. WSDL [2] 1.0 failed to provide the ability to augment the language with our own constructions and it was not expressive enough to describe and represent a variety of types of service interface patterns and invocation patterns. It did not have sufficient abstraction to accommodate bindings to a variety of different endpoints via diverse communication protocols. Since NEMO is intended to support models were intended to support, we did not want to tie our architecture completely to web service standards and decided to create a framework that could leverage web service standards but is not dependent on them. Thus, we initially created our service description language called NSDL that emphasized polymorphic data types, allowing us to describe more cleanly the services functionality that we require. With WSDL 1.1 and anticipating 1.2, we get sufficient expressivity and extensibility to meet our needs.

We define a profile to be a set of thematically related data types and interfaces defined in WSDL for the NEMO framework. We currently have two profile categories: "Core", which includes the foundational set of data types and service messages necessary to support fundamental NEMO framework interaction patterns and infrastructural functionality and "DRM" which describes the Digital Rights Management services that can be realized with NEMO. Many of the types and services defined in these profiles are abstract and need to be specialized before they are used. Other profiles are built on top of the Core profile. Some of the profiles we have currently defined, based on the above categories, include:

- Core Profile - the profile from which all other profiles are based. It includes a basic set of generic types that serve as the basis for creating more complex types.
- Core Profile Extension – the primary specialization of the types in the Core profile to concrete representations.
- Core Service Profile – the profile that defines a set of general infrastructure services. The service definitions are abstract and must be specialized.
- Core Service Profile Extension – the primary specialization of the services defined in the Core Service Profile to concrete representations.
- DRM Profile – the profile from which all DRM-related profiles are based. It includes a set of generic types that serve as the basis for creating more complex DRM-specific types.
- DRM Profile Extension – the primary specialization of the types in the DRM profile to concrete representations.
- DRM Service Profile – defines a set of general DRM services.
- Octopus DRM Profile – a further specialization of DRM services defined for the Octopus DRM [10] System. Octopus is a lightweight DRM client created within InterTrust. Octopus also introduces some new types and further extends certain types specified in Core Profile Extensions.

Some services defined in the Core profile include:

- Authorization – services related to authorization of a participant (such as node) to use a resource

(service).

- Peer Discovery – services related to the discovery of NEMO nodes.
- Notification – services related to the delivery of targeted messages to a given set of nodes.
- Service Discovery – services related to the discovery of services provided by some set of service providing nodes.

Some basic services defined in the DRM profile include:

- Provisioning – services for obtaining the necessary credentials, policy, and other objects necessary for a CE device, software application, etc to participate within a specific context that uses DRM.
- Licensing – services for obtaining DRM licenses.
- Membership – services for obtaining objects that establish membership within some designated domain.

We factored the NEMO profiles into a set of Generic Interface Specifications (describing an abstract set of services, communication patterns, and operations), Type Specifications (containing the data types defined in the NEMO profiles), and Concrete Specifications (mapping abstract service interfaces to concrete ones including bindings to specific protocols).

5. Architectural overview

The following section goes into more detail on some of the concepts outline in the logical model section.

The Service Adaptation Layer provides a common way for service providers to expose services, process requests and responses, and orchestrate services in the NEMO framework. It is the logical point at which services are published. It provides a foundation on which to implement other specific service interface bindings. One may associate with the description a list of one or more service providers responsible for authorizing access to the service, along with a description of any necessary orchestration that must take place between endpoints.

Typically, an implementation of the Service Adaptation Layer consists of the following layered elements:

- A layer encapsulating the service interface entry points described in the WSDL [2] bindings of the service interface. Through these access points, one invokes services, passes parameter data, and collects results.
- A layer that corresponds to the logic for message processing. This typically contains some sort of message pump that drives the processing, some type of XML data binding support, and low level XML parser and data representation support.
- A layer representing the native services available (onto which the corresponding service messages are mapped).

The Framework SDK provides a collection of optional support functions making it easier to enable an entity to participate in the NEMO framework. There may be multiple versions of the support library supporting different functionality, e.g. client application functionality versus functionality needed by service providers. In the general case, the functionality available within the support library includes:

- Service Access Point – encapsulates the functionality necessary to allow a NEMO node to make a service invocation request to a targeted set of service providing nodes and receive a set of responses.
- Workflow Collator – service provider interface allowing a node to manage and process collections of messages. It provides the basic building blocks to orchestrate services. Currently, this interface is most often implemented by a node that supports message routing and supports the intermediate queuing and collating of messages.
- Miscellaneous Support Functionality – routines for generating message ids, timestamps, manipulating XML message parts etc.

Service Access Point (SAP) NEMO-enabled nodes may use diverse discovery, name resolution, and transport protocols. This necessitates the creation of a flexible and extensible communication API.

The framework must also support diverse communication styles such as synchronous or asynchronous RPC as well as styles supporting one-way interface invocation and client callbacks. The SAP API provides a common interface to the functionality necessary for allowing a NEMO node to make service requests and receive responses for a set of targeted NEMO nodes. A SAP may be implemented in a variety of forms within the boundaries of a client (in the form of a shared library) or outside the boundaries of the client (in the form of an agent running in a different process). One tailors the exact form of the SAP implementation toward the needs of the platform or client. Use of the SAP is optional, although in general it provides significant utility. A SAP can be used as a common, reusable API for service invocation. It can encapsulate the negotiation and use of a transport channel. For example, some transport channels may require SSL session setup over TCP/IP, some channels may only support unreliable communication over UDP/IP, and others may not be IP based at all. A SAP can

encapsulate the discovery of an initial set of NEMO nodes for message routing. For example, a cable set-top box may have a dedicated connection to the network and mandate that all messages flow through a specific route and intermediary. A portable media player in a home network may use UPnP discovery to find multiple nodes that are directly accessible. Clients may not be able or may not choose to converse directly with other NEMO nodes by the exchange of XML messages. In this case, a version of the SAP may be created that exposes and uses whatever native interface and protocols the client supports.

The SAP pattern supports two common communication models 1) *Message-based* – where the SAP forms XML request messages and directly exchanges NEMO described messages with the service provider via some interface binding, *and 2) Native* - where the SAP may interact through some native communication protocol with the service provider. The Service Access Point internally may translate to/from XML messages defined in the framework.

In addition to these basic models patterns it is possible to have implementations of the SAP that implement combinations of the above patterns or new patterns

The Workflow Collator (WFC) helps fulfill most non-trivial NEMO service requests by coordinating the flow of events of a request, managing any associated data including transient and intermediate results, and enforcing the rules associated with fulfillment. Examples of this type of functionality can be seen in the form of transaction coordinators ranging from simple transaction monitors in relational databases to more generalized monitors as seen in Microsoft MTS/COM+. The Workflow Collator is a programmable mechanism through which NEMO nodes orchestrate the processing and fulfillment of service invocations.

The WFC can be tailored towards a specific NEMO node's characteristics and requirements, and is designed to support a variety of functionality ranging from traditional message queues to more sophisticated distributed transaction coordinators. The simplest form of the WFC provides an interface for storage and retrieval of arbitrary service-related messages. By building on this, it is possible to support a wide variety of functionality including:

- Collection of service requests for more effective processing.
- Simple aggregation of service responses into a composite response.
- Manual orchestration of multiple service requests and service responses in order to create a composite service.
- Automated orchestration of multiple service requests and service responses in order to create a composite service.

The basic service interaction pattern begins with a service request arriving at some NEMO node accepted through the node's Service Adaptation Layer. The message is handed off to the WSDL Message Pump that will initially drive and in turn be driven by the WFC to fulfill the request and return a response. In even more complex scenarios, the fulfillment of a service request will require the participation of multiple nodes in a coordinated fashion. The rules for processing the request may be expressed in the framework's service description language or in other service orchestration description standards such as BPEL4WS [3].

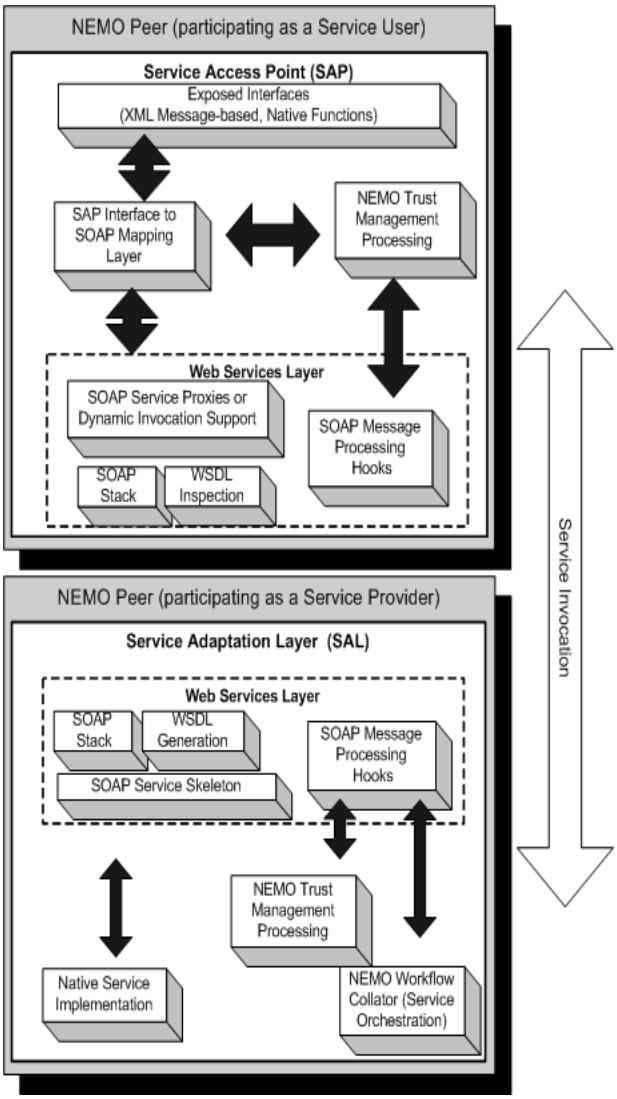
When a message is given to the WFC, it must determine the correct rules for processing this request. Depending upon the implementation of the WFC, the service description logic may be represented in the form of a fixed state machine for a set of services that the node exposes or it may represented in ways that support the processing of a more free form expression of the service processing logic. The WFC architecture is modular and extensible supporting plug-ins. The framework itself supports a simple way of expressing the rules for orchestrating multiple NEMO services into composite services. Part of our ongoing work involves extending this mechanism and determining how to make it synergistic with other forms of service orchestration.

In addition to interpreting service composition and processing rules, the WFC will need to determine whether to use the NEMO message in the context of initiating a service fulfillment processing lifecycle or as input in the chain of an ongoing transaction. NEMO messages include IDs and metadata that are used to make these types of determination; it is also possible to extend NEMO messages to include additional information that may be service transaction specific, facilitating the processing of messages.

Notification Services are directly supported by the NEMO Framework. A notification represents a message targeted at interested NEMO-enabled nodes received on designated service interface for processing. Notifications may carry a diverse set of payload types for conveying information and the criteria used to determine if a node is interested in a notification is extensible including identity-based as well as event-based criteria.

The following diagram depicts the composition of a simple peer that uses a NEMO SAP for interacting with a peer that is a NEMO-enabled service provider. In this example, web service protocols and standards are used for exposing services as well as for transport.

NEMO TM leverages WS-Security in this example using credentials conveyed in SOAP headers.



6. Authorization

Before any NEMO node allows access to a specified service, it must first determine if the requesting node is permitted access to that service and under what conditions. Access permission is based on a trust context for interactions between service requestor and service provider and policies permitting the requesting node access to any of the requested service's interfaces.

Establishing Trust

The NEMO framework does not mandate the specific requirements, criteria, or decision making logic for how an arbitrary set of nodes come to trust each other. Trust semantics may vary radically from node to node. The NEMO framework instead strives to provide a standard set of facilities that allow nodes to negotiate a mutually acceptable trusted relationship. In the determination and establishment of trust between nodes, the NEMO framework supports the exchange of information (including evidence and policy) between nodes, which can be used for establishing a trusted context. An example of this includes a service provider conveying some policy associated with the use of a service that a requestor must satisfy by the presentation of some evidence.

Within the NEMO framework, information used in the establishment of a trusted context may be exchanged using Service-Binding Properties where a trust context is established as part of the service interface binding. For example, if a node can expose its service in the form on an HTTP Post over SSL or as a Web Service that requires WS-Security XML Signature. Alternatively, one can exchange policy and evidence using request and response messages. The NEMO Framework also supports combinations of these approaches. For example a communication channel associated with a semi-trusted service binding may be used to bootstrap the exchange of other security-related credentials more directly or exchanging security-related credentials (which may have some type of inherent integrity) directly and using them for the establishment of a secure communication channel associated with some service interface binding.

Policy Managed Access

In addition to establishing a trusted context between a set of interacting nodes, before a service providing node allows a requesting node to access a resource it must also determine whether this access is permitted based on any policy associated with the resource. The policy decision mechanism used here may be local and/or private. However, the NEMO Framework provides a consistent, flexible mechanism for supporting this functionality. As part of the service description, one can designate specific NEMO nodes as authorization service providers. An authorization service providing node must implement a standard service for handling and responding to "Authorization" query requests. Before access is allowed to a service interface, the targeted service provider will dispatch an "Authorization" query request to any authorizing nodes for its service, and access will only be allowed one or more nodes respond indicating that access is permitted. Essentially the Authorization service allows any NEMO node to participate in the role of policy decision point (PDP). NEMO is policy management system neutral; it does not mandate how an authorizing node reaches decisions about authorizations to services based on an authorization query.

While NEMO does not mandate the specific policy

engine or policy language used by an authorizing node for a given platform, for interoperability NEMO currently uses SAML [8] to encapsulate assertions containing evidence and policy assertions, regardless of the language used.

7. Roles of Participation

One of the goals of the NEMO framework is to enable, promote, and actively support a model where any peer can spontaneously offer a variety of functionality by exposing services. The framework actively discourages viewing peers as having a fixed set of capabilities but instead encourages a model where a peer at any point in time is participant in one or roles.

A role is defined by a set of services that a given peer exposes in combination with a specific behavior pattern. At any given moment a NEMO-enabled node may act in multiple roles based on a variety of factors: its actual implementation footprint providing the functionality for supporting a given set of services, administrative configuration, information declaring the service the peer is capable of exposing, and load and runtime policy on service interfaces.

Currently in the NEMO framework we have not yet tried to define a definitive set of roles based on service type groupings. Over time, as common patterns of participation are determined and as new services are introduced, we may need to define a formal role categorization scheme. However based on existing functionality and observed patterns we have defined a preliminary set of roles that may be formalized over time. These include:

Client - this is the simplest role where no services are exposed and the peer simply uses services of other peers.

Authorizer - this role denotes a peer acting as a Policy Decision Point (PDP) determining if the requesting principal has access to a specified resource with a given set of pre-conditions and post-conditions.

Gateway - in certain situations a peer may not be able to directly discover or interact with other service providers, for reasons including: transport protocol incompatibility, inability to negotiate a trusted context, or lack of the processing capability to create and process the necessary messages associated with a given service. This role denotes a peer acting as a bridge to another peer in order to allow the peer to interact with a service provider. From the perspective of identity and establishing an authorized and trusted context for operation, the requesting peer may actually delegate to the gateway peer its identity and allow that peer to negotiate and make decisions on its behalf. Alternatively, the gateway peer may act as a simple relay point forwarding or routing requests and responses.

Orchestrator - in situations where interaction with a set of service providers involves some type of non-trivial coordination of services possibly including transactions, distributed state management, etc, it may be beyond a peer's capability to participate in such a scenario. The role is a specialization of the Gateway role. A peer requests an orchestrator peer to act in its behalf intervening with one or more services. The orchestrating peer may use certain additional NEMO components such as an appropriately configured Workflow Collator in order to satisfy the orchestration requirements.

8. Consumer Media Applications

Since our ultimate goal is to enable the oft-repeated "instant gratification of request for any media, in any format, from any source, to any place, at anytime, on any device complying with any agreeable set of usage rules", we developed an informal model that helps us demonstrate how we use NEMO to achieve this goal. This model helped us in the separation of concerns process in system architecture discovery. We explain only the highest level of the model, and then, without enumerating every aspect of how NEMO enables all of the media services that one can imagine, we show how NEMO allows low level services from different tiers in the model to be assembled into richer end-to-end media services

8.1 Media distribution model

In this model there are four tiers of service components: 1) Content Authoring, Assembly, and Packaging services, 2) Web-based Content Aggregation and Distribution services, 3) Home Gateway services, and 4) Consumer Electronics devices.

Each of these four tiers clearly has significantly different requirements for security, rights management, service discovery, service orchestration, user interface complexity, and other service attributes. The first two tiers fit very roughly into the models that we see for "traditional" web services, while the last two tiers fit more into what we might call a personal logical network model, with certain services of the home gateway being at the nexus between the two types of models. However, services of CE devices could occasionally appear in any of the tiers. Thus, we have the dilemma where we want to specialize parts of the framework for efficiency of implementation, while

being general enough to encompass an end-to-end solution. For relatively static and centralized web services, a UDDI directory and discovery approach may work well, but for a more dynamic transient merging of personal networks, discovery models such as found in UPnP and Rendezvous are more appropriate. Thus, we need to be able to include multiple discovery standards in our framework. When rights management. is used for media distribution through wholesale, aggregator, and retail distribution subtiers, there can be many different types of complex rights and obligations that need to be expressed and tracked. This requires a highly expressive and complex rights language, sophisticated content governance and clearing services, and a global trust model. However, rights management and content governance for the home gateway and CE device tier generally requires a different trust model and needs to emphasize fair use rights that are straightforward from the consumer's point of view. Peer devices in a personal logical network want to interact using the simple trust model of that network, and they need to interact with peers across a wide area network using a global trust model perhaps through proxy gateway services.

At the consumer end, complexity arises from automated management of content availability across devices, some of which are mobile and intermittently intersect multiple networks. Thus, our approach to rights management, while enabling end-to-end distribution, is heterogeneous, supporting a variety of rights management services, including services that interpret distribution rights expressions and translate them, in context, to individual consumer fair use rights in a transaction that is orchestrated with a sales transaction or another event where a subscription right is exercised.

8.2 NEMO solutions

We are currently using NEMO to link various consumer devices to a number of different services in the multi-tiered environment described above. We have successfully demonstrated interoperability in one interconnected system using cell phones, game platforms, PDAs, PCs, web-based content services, discovery services, notification services, and update services. We support multiple media formats [MP4, WMF, et al], multiple discovery protocols (for discovery of services over Bluetooth and through registries such as UDDI, LDAP, Active Directory), and IP-based notification and Wireless SMS notification on the same device. We use the orchestration feature to help the consumer overcome interoperability barriers. When there is a query for content, the various services are coordinated in order to fulfill the request, including, discovery, search, matching, update, rights exchange, and notification services. We are attempting to converge on a state where a consumer can use most any device, make a wish for content, and be instantly fulfilled with the content and the rights and rendering capabilities (within obvious limits of the hardware) to both use and share the content. The orchestration capability allows the consumer to view all home and internet-based content caches from any device at any point in a dynamic multi-tiered network. We are extending this capability to more advanced services that promote sharing of streams and play lists, making impromptu broadcasts and narrowcasts easy to discover and connect to, using many different devices, while ensuring rights are respected.

Beyond the consumer-centric side, we are looking at ways to provide an end-to-end interoperable media distribution system that does not rely on a single set of standards for media format, rights management, and fulfillment protocols. In the value chain that includes content originators, distributors, retailers, service providers, device manufacturers, and consumers, there are a number of localized needs in each segment. This is especially true in the case of rights management, where content originators need to express rights of use that may apply differently in various contexts to different downstream value chain elements. A consumer gateway has a much more narrow set of concerns, and an end user device has a yet simpler set of concerns, namely just playing the content.

With a sufficiently automated system of dynamically self-configuring distribution services, content originators can produce and package content, express rights, and confidently rely on value added by other service providers to instantly provide the content to all interested consumers, no matter where they are or what kind of device they are using. We use NEMO to fulfill this goal and provide means for multiple service providers to innovate and introduce new services that benefit both consumers and service providers without having to wait for or depend on a monolithic set of end-to-end standards. This approach allows digital rights management to be decomposed into components with a more natural separation of concerns that focus on policy management of service interfaces rather than on copy protection. This has the potential to change the tension between consumers and content providers in the digital content domain as the NEMO enriched infrastructure provides consumers with better information, more useful services and instant gratification. Policy management can limit the extent to which pirates can leverage those legitimate services. NEMO allows the network effect to encourage the evolution of a very rich set of legitimate services providing better value than pirates can provide.

Here is a simple example that we built using NEMO. It illustrates: bridging of discovery Services - UPnP based service discovery as a mechanism in finding and linking to a UDDI based service, service interactions between Personal Area Network (PAN) and Wide Area Network (WAN) services, negotiation of a trusted context for service use, and provisioning of a new device for DRM.

A consumer, having bought a new music player, tries to play a DRM protected song. The player can support this DRM but needs to be personalized. The Player is wireless, supports the UPnP and Bluetooth protocols, and has a set of X.509 [14] certificates it can use for purposes of validating signatures and signing messages. The player is NEMO-enabled in that it can form and process a limited number of NEMO service messages but it does not contain a NEMO SAP because of resource constraints.

The Player is able to participate in a Personal Area Network (PAN) in the users home, that contains a NEMO enabled, internet connected, home gateway device with Bluetooth and a NEMO SAP. Both the Player and the Gateway‘s UPnP stack have been extended to support a new service profile type for a “NEMO-enabled Gateway” service.

The consumer downloads a song and tries to play the content. The player determines it needs to be personalized and begins the process. The Player initiates a UPnP service request (M-Search) for a NEMO gateway on the PAN.

The player finds a NEMO gateway service, and the gateway returns the necessary information for allowing the Player to connect to the service. The Player forms a NEMO Personalization request message and sends it to the gateway service. The request contains an X509 certificate associated with the device’s identity.

The Gateway upon receiving the request determines it cannot fulfill the request locally but has the ability to discover other potential service providers. However, the gateway has a policy that all messages it receives must be digitally signed and thus it rejects the request and returns an authorization failure stating the policy associated with processing this type of request.

The Player upon receiving this rejection notes the reason for the denial of service and then re-submits the request to the Gateway with the request digitally signed. The Gateway accepts the message.

As previously mentioned, the Gateway cannot fulfill the request locally but can perform service discovery. The Gateway is unaware of the specific discovery protocols its SAP implementation supports and composes a general attribute-based service discovery request based on the type of service being desired and dispatches the request via the SAP. The SAP, configured with the necessary information to talk to UDDI registries, upon receiving the request converts it into a native UDDI query of the appropriate form and makes the query. The UDDI registry knows of a service provider that supports DRM personalization and returns the query results. The SAP receives these results and returns an appropriate response, with the necessary service provider information, in the proper format, to the gateway. The gateway extracts the service provider information from the service discovery response and composes a new request for Personalization based on the initial request on behalf of the Player.

The request is submitted to the SAP. In this case there is a requirement for communication with a Personalization service that exposes its service through a web service described in WSDL referenced in the service interface description, the SAP invokes the service and obtains the response.

The Gateway then returns the response to the Player, which can use the payload of the response to personalize its DRM engine.

At this point, the music player is provisioned, and can fully participate in a variety of local and global consumer oriented services. These can provide full visibility into and access to a variety of local and remote content services, lookup, matching and licensing services, and additional automated provisioning services all cooperating in the service of the consumer. For example, a consumer using a personal media player at home can enjoy the simplicity of a CE device, but leverage the services provided by both gateway and peer devices. When the consumer travels to another venue, then the device can rediscover and use most or all of the services available at home, and through new gateway services be logically connected to the home network, while enjoying the services available at the new venue that are permitted according to the various policies associated with those services. Conversely, the consumer’s device can provide services to peers found at the new venue.

9. Conclusion

NEMO is an experimental system that we will make more widely available as a reference design in early 2004. Since the policy management aspects of a system such as this are of great interest, we are focusing on those and related aspects. We created our instance of NEMO using a set of distributed policy management services based on InterTrust’s PolicyWorks [9] and the PolicySpeak policy expression language, while using other standards such as SAML, WS-Security, and WS-Policy [16]. However, NEMO can use other policy management frameworks and languages such as IBM’s

Policy Director [7] and XACML [6]. Indeed, the framework is able to accommodate simultaneously multiple approaches to P2P trust management. In the area of open P2P service orchestration, the issue of transitivity of trust is challenging. We expect to spend more time on this issue in the future.

We believe that we have developed an approach to media services that allows rights management to be more naturally distributed in favor of a variety of stakeholders, to be supportive of consumer fair use, and to enable more convenient and immediate access to media. While the framework is ambitious, we designed it to accommodate contributions from many sources outside of our research group, and we look forward to receiving and working with those contributions.

10. References

[1] Scott Werden (WRQ), Colleen Evans (Sonic Software), Marc Goodner (SAP), WS-I Usage Scenarios, Web Services Interoperability Organization (WS-I), http://www.ws-i.org/

[2] Erik Christensen (Microsoft) Francisco Curbera (IBM Research), Greg Meredith (Microsoft), Sanjiva Weerawarana (IBM Research), Web Services Description Language (WSDL) 1.1 W3C Note 15 March 2001, http://www.w3.org/TR/wsdl

[3] Francisco Curbera (IBM), Yaron Goland (BEA Systems), Johannes Klein (Microsoft), Frank Leymann, (IBM), Dieter Roller, (IBM), Satish Thatte (Microsoft), Sanjiva Weerawarana (IBM), BPEL/BPEL4WS - Business Process Execution Language for Web Services, Version 1.0 31 July 2002, http://www.ibm.com/developerworks/library/ws-bpel/

[4] Apple/Darwin Group, Rendezvous, http://developer.apple.com/darwin/projects/rendezvous/

[5] UPnP Forum, Basic Device V 1.0 Profile, MediaServer V 1.0 and MediaRenderer V 1.0 Profile, Internet Gateway Device (IGD) V 1.0 Profile, http://www.upnp.org/

[6] OASIS - Editors: Simon Godik (Overxeer), Tim Moses (Entrust), eXtensible Access Control Markup Language (XACML) Version 1.0, OASIS Standard, 18 February 2003, http://www.oasis-open.org/

[7] IBM, IBM Policy Director - IBM WebSphere Application Server, Version 3.0 Enterprise Edition, http://www.ibm.com

[8] OASIS - Editors: Phillip Hallam-Baker (VeriSign), Eve Maler (Sun Microsystems), Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) OASIS Standard, 5 November 2002, http://www.oasis-open.org/

[9] InterTrust, PolicyWorks – A Platform for Enterprise Policy Management, June 2002 , Internal Memo

[10] InterTrust, Octopus Principles of Operation, August 2003 , Internal Memo

[11] OASIS - Editors: Tom Bellwood (IBM), Universal Description, Discovery and Integration (UDDI) V2, OASIS Standard, 19 July 2002, http://www.uddi.org/specification.html

[12] Bill Joy and Jim Waldo (Sun Microsystems), Jini Network Technology, March 1999 http://wwws.sun.com/software/jini/

[13] Editors: Chris Kaler (Microsoft), Web Services Security (WS-Security), Version 1.0, 5 April 2002, http://www-106.ibm.com/developerworks/webservices/library/ws-secure/

[14] Santesson, et al, "Internet X.509 Public Key Infrastructure Qualified Certificates Profile," http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-X.509-200003-I

[15] Bluetooth.org, Bluetooth Core Specification, 1.0B January 2003 , https://www.bluetooth.org/docman2/ViewProperties.php?group_id=53&document_content_id=330

[16] Editors: Maryann Hondo (IBM), Chris Kaler, (Microsoft), Web Services Policy Framework (WSPolicy), 28 May 2003, http://www-106.ibm.com/developerworks/library/ws-polfram/