Multi-Tier Diversified Architecture for the Next Generation Internet

Subharthi Paul
Washington University in St. Louis
Department of CSE
One Brookings Drive,
Saint Louis, MO 63130
spaul@wustl.edu

Raj Jain
Washington University in St. Louis
Department of CSE
One Brookings Drive,
Saint Louis, MO 63130
jain@cse.wustl.edu

Jianli Pan
Washington University in St. Louis
Department of CSE
One Brookings Drive,
Saint Louis, MO 63130
jp10@cse.wustl.edu

ABSTRACT

We propose a next generation Internet architecture that will allow natural sharing of resources among multiple organizations by dynamically reconfiguring and creating a virtual network for a particular application. Our architecture called "Internet 3.0" consists of a 3-tier object model. The bottom tier consists of a high-speed network infrastructure owned by multiple ISPs. The second tier consists of hosts owned by different organizations such as DoE, DARPA, Amazon, etc. The third tier consists of users and data objects. This is a three tiered virtualization model as compared to single tier virtualization being discussed in NSF's GENI and FIND communities. This three tiered virtualization model allows users to quickly setup a virtual cloud for any application. The users and data can easily move among the host clouds that themselves move on the infrastructure cloud. The users, data, hosts, and infrastructures are owned by different organizations that have their own policies for sharing and isolation. Also the multi-tiered virtualization concept allows each tier to recursively provide richer and more diversified set of core services to the tier above it.

Keywords

Next Generation Internet Architecture, Overarching Architecture, Application Specific Network Architecture, Cloud Computing, Distributed Application Contexts, Multi-Tier, Policy Framework, Object Abstraction, Quality of Service.

1. INTRODUCTION

The current Internet designed around the modest requirements of file transfer and resource sharing applications fail to satisfy the diverse needs of modern distributed applications. Also, the context of "networking" has evolved significantly over the last 40 years of our experience with network systems, making it necessary to retrospect on the basic underlying design of the present Internet.

The trend in distributed computing is emerging towards running applications on leased resources from third party compute resource providers such as cloud platforms. The wide scale availability of compute resources consolidated over multiple cloud computing platforms ranging from private clouds within universities and research organizations, science clouds consisting of various grid resources such as supercomputing centers etc, and commercial clouds such as Amazon, GoGrid, Rackspace, etc, provide a highly distributed and diverse platform for the deployment of disruptive, novel and distributed application contexts. Clouds, with much better and flexible management plane than original grids have allowed them to better utilize their resources through more dynamic and efficient of sharing among multiple different contexts. Also, isolation handled through virtualization allow strong performance guarantees. These have allowed cloud platforms to build a viable business model for leasing compute resources at extremely cheap prices and at high temporal granularity. However, this evolution of distributed computing (in terms of diversity of distributed resource availability) is unfortunately not matched by the underlying wide area networking substrate of the Internet. Thus, applications deployed over the cloud are currently restricted to being stand alone instances over compute resources leased from a single cloud provider that elastically serve the resource requirements for variable load conditions. The key motivation of this paper is to evolve a next generation Internet architecture that allows the Internet to match the service diversity and performance guarantees of cloud platforms, and thus, serve the requirements of these highly distributed application contexts with diverse requirements, more naturally.

One of the key factors restricting the Internet's service diversity is that although the Internet was built over a fundamentally novel packet switching paradigm as against the circuit switching paradigm of telephone networks, it still preserved the same unicast conversational model of telephony as its basic connectivity primitive. The reason for choosing this primitive in telephony was understandably to evolve the underlying connectivity of the system separately from the purpose for which it would eventually be used for. While, this model worked well for a paradigm which was restricted to a single application context, it has proved to be extremely restrictive for the more diversified set of application contexts of the Internet. The reason being that this conversational model does not necessarily map into the diversity of the application contexts that can be supported over the highly flexible packet-switching primitive of the underlying network. Thus, the Internet 3.0 architecture is based on the

CCV Conference 2010, May 17–18, 2010, Singapore.

This work was supported in part by a grant from Intel Corporation and NSF CISE Grant #1019119.

concept of a "communication-paradigm," contrary to the existing "communication-system" based architecture of the present Internet. The difference between these two architectural approaches being that in the "communication-system" based architecture the underlying communication primitives evolve largely independent of the specific needs of the application context that is installed over it, while the "communication-paradigm" based architecture ideally allows the communication primitives to evolve per the specific requirements of application contexts . The "communication paradigm-based" architecture is driven by the better optimization scope of "verticlized" designs over modular designs. However, defining such an architecture is non–trivial and we are faced with the same classical trade-off between simplicity (offering scalability) and flexibility (introducing complexity) that drove the architectural design choices behind the present Internet's *single best-effort* connectivity service model.

Additional complexities arise from the need to embed an explicit policy framework into the communication paradigm. The truly multi-ownership nature of modern application contexts necessitate the ability to express policies at the required level of granularity, within a framework that allows explicit policy negotiation and enforcement. To motivate our case further, consider an example of an enterprise application running over compute resources leased from multiple cloud providers, and each distributed location connected over the multi-domain infrastructure of the Internet. Multiple such distributed contexts consolidated over multi-owner substrate of compute and connectivity infrastructure resources demand strong isolation guarantees in terms of both performance and security. In the absence of a standardized policy platform, interactions across ownership boundaries shall try to implement local policies in ad-hoc, non-standard ways making the system in-efficient, intractable and possibly in-deterministic. An example can be seen in the state of policy implementation in the current wide area inter-domain routing plane. BGP[21], the *de-facto* inter-domain routing protocol of the Internet, tried to work around the lack of an explicit policy framework in inter-domain routing (for historic reasons), by allowing autonomous systems to specify their local transit policies by conflating them with the forwarding behavior of the AS. This conflation of policy and functionality in the routing plane of the current Internet results in local AS policies having global impact on the availability of end-to-end paths. This in turn, indirectly impacts routing diversity, resilience, efficiency, availability, path quality and multiple other performance and quality metrics. The policy impact (or lack of it) on modern applications, within the complex distributed ownership context over which they are deployed, is inestimable. However, the true challenge is to develop this policy framework within manageable limits of complexity for it to be acceptable. Thus, while policy expressiveness and granularity shall dictate the available diversity to define an application context, abstraction hierarchies to bound the complexity of these policy interactions also need to be defined.

The discussion thus far, leads to the two architectural basis of the Internet 3.0 architecture – 1. ***The Three-Tier Object Model***, and 2) ***The Object Abstraction***. The "three-tier object model" realizes the inherent tiered nature of communication paradigms and allows an explicit separation of data/users, hosts and infrastructure, establishing them as independent entities within a tiered policy framework. The "object abstraction" is the glue between the functional plane and the policy plane of the Internet 3.0 architecture. It defines the most primitive building block the Internet 3.0 architecture that allows specific application contexts to be dynamically composed (Functional Plane) over distributed resources leased from multiple resource owners through an explicit policy-negotiation mechanism (Policy Plane). In the rest of the paper, we shall discuss these architectural artifacts of the Internet 3.0 architecture in more detail and try to justify the design choices through well-founded design principles.

2. DESIGN PHILSOPHY

The Internet 3.0 architecture is governed by the following two design philosophies:

A ***Diversity naturally follows ability to express and enforce policy at the required level of granularity:*** Mostly, although diversity is in the offing, choices can not be enabled owing to the lack of a proper policy framework allowing policy expression, enforcement and negotiations. As an example, years of research and multiple technically sound solutions later, QoS routing could not be widely deployed over the Internet. The reason can be traced to the lack of a proper business framework wherein multiple autonomous systems could negotiate their individual services and aggregate them to provide an end-to-end inter-domain QoS routing service to applications that need it. Also, the extent of diversity depends directly on the level of granularity of policy enforcements. An example can be cited in the per-flow and flow-class managements of Inteserv[1] and Diffserv[16], respectively.

B. ***True diversity can be achieved only through the explicit separation of policy from functionality:*** An example in support of this design philosophy may be seen in current Internet inter-domain routing where individual autonomous system (AS) policies can effect the global state of the distributed routing algorithm [7][8][9]. AS relationships govern routing quality. The reason can be attributed to the conflation of the routing state to represent both, reachability information as well as AS level policies.

The three-tier object model recursively applies these design philosophies to each of the three tiers of user/data, hosts and infrastructure. While the three-tier object model builds the policy framework of the Internet 3.0 architecture, the "object abstraction" provides the basic primitive for realizing the diversity requirements of specific application contexts within this policy framework.

The Internet 3.0 architecture implements "**Functional Plane"** overlaid with a **"Policy Plane."** The "functional plane" realizes the functional diversity requirements of specific application contexts over a set of ownership and policy driven primitives exposed through the "policy plane." The two planes are glued together using the basic abstraction of "*objects*" as shall be discussed later in this paper.

3. THE POLICY FRAMEWORK

Building further on the discussion in Section 1 on "communication-system" based design and "communication-paradigm" based design, the key realization is that the application context of modern distributed applications is inherently tiered. The "communication-system" based design of the current Internet is built on the basic primitive of "connectivity" between two infrastructure access identifiers (in this case IP addresses). The

use-context of this "communication-system" (in this case data communication) is a *side-effect* of this basic connectivity premise. Similarly, proponents of data-centric networks [13] argue in favor of re-defining the "communication-system" such that it is built with "data connectivity" as its fundamental primitive. This new connectivity premise more suitably models the data-communication *use-context* of the Internet. However, it too makes some very simplifying assumptions on the nature of the communication paradigm. *Firstly,* it replaces the "where" based "communication system" model with a "what" based model under the assumption that the explicit representation of "hosts" and "topological locations" would not be necessary for any application context. *Secondly*, it neglects the fact that the multi-ownership policy fabric shall still govern the implementation of the "connectivity" primitive in non-standard ways (and beyond the control of the primitive itself) in the absence of an explicit policy framework.

The purpose of this discussion is to drive the point that the future Internet architecture needs to define a framework that allows multiple diversified contexts to co-exist. Also, such functional diversity needs to be well aligned with the policy fabric of multiple ownerships through explicit policy expression, negotiation and enforcement.

The generality of the proposed framework is based on the *tiered dependency diagram* (discussed next in Section 3.1) on *entities* and their ownerships are represented through *realms* (Section 3.3).

3.1 Entities

Data, host and infrastructure represent entities. Entities are broad classification of resource types. Any networked system is implicitly organized as an interaction between these entities. However, the conflated design of the current Internet neither allows these interactions to be explicitly associated with the individual entities nor allows these entities to specifically enforce their policies. The core of the multi-tier diversification architecture lies in making inter-entity interactions explicit and design a framework for active negotiation of policies between them. "Tiered entity-dependency diagram" (Figure 1) depicts an obvious dependency relationship between these entities in a networking context. The bottom tier consists of a high-speed network infrastructure. The second tier consists of hosts. The third tier consists of Data. The direction of the arrows implicitly represents a "depends on" relationship. Also, the dependencies are transitive, meaning "Data needs to reside on a Host connected through an infrastructure point of presence" to participate in a valid networking context. Dependency among the entities represents "fundamental constraints" in the architecture that guide the process of dynamic composition of "objects" to form a valid requirement specific networking context.

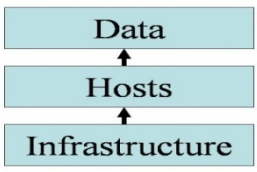

Figure 1. Tiered Entity Dependency Diagram

3.2 Conflated Identities

The current Internet design suffers from the problem of conflated identities. As a direct consequence of the "communication-system" based architecture of the current Internet (as discussed above), a single identifier space (IP addresses in the case of current Internet) over which the "connectivity" primitive is defined, is conflated to represent the tiered-communication paradigm. De-conflated identities are desirable primarily for two reasons:

A. Functional Diversity: The tiered-paradigm of modern distributed application contexts based on the "*tiered entity dependency diagram*" primarily requires independent identifier space for each tier to avoid strong-coupling between functions defined in each tier. As an example, the contextual overloading of IP addresses to serve both as a routing locator as well as a connection identifier for end-to-end TCP connections, creates unnatural dependencies which prevent simple and optimal implementation of host mobility, multihoming and site traffic-engineering functions.

B. Policy Granularity: While identifiers need to be de-conflated along the tiers of the "*tiered entity dependence diagram*" for functional independence across the tiers, the identity space must also be de-conflated to represent the ownership boundaries along the different tiers. This is especially true in the context of modern distributed applications where resources along the multiple entity tiers may be leased from multiple resource providers such as infrastructure resources from ISPs, compute resources from multiple cloud platforms and data resources from multiple content providers, and composed into a single application context. Thus, the identifier space needs to be able to represent the required level of granularity of ownership-dictated policy expression, control and enforcement to be aligned with the required functional diversity of the architecture.

3.3 Realms

Realms overlay entities with a discreet ownership framework (Figure 2). Ownership entails related administrative and management responsibilities. In the "tiered entity-dependency diagram" (Figure 1), the bottom tier infrastructure is owned by multiple infrastructure owners. The second tier of hosts is owned by individual users or different organizations such as DoE, DARPA, Amazon, etc. The third tier of users and data may belong to specific organizations or individual users. Thus, realms represent logical divide of entities into multiple ownership and management domains.

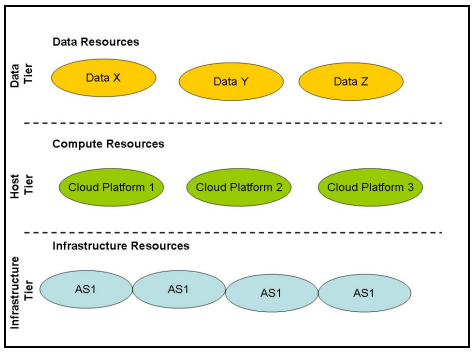

Figure 2. Realms

3.4 Objects

An object is a logical instantiation of an entity, in a specific networking context. Objects encapsulate the complexities of resource allocation, resource sharing, policy enforcements etc and expose a standard interface representing capabilities (in terms of standardized abstract parameters) and fixed or negotiable policies. Objects are owned and managed by realms and represent the responsibilities and policies pertaining to its realm membership.

3.5 Three-Tier Object Model

Finally, Figure 3 represents the three-tier object model that maps the functional diversity requirements of modern application contexts to the policy enforcement granularity of the multi-ownership resource base that compose these diversified contexts.

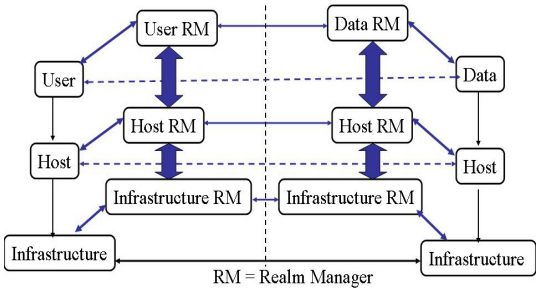

Figure 3. Three-Tier Object model

Object represents a fundamental service unit managed and leased by the owner "realm." The "objects" in each tier and belonging to each realm are accessible through an "object abstraction" interface. We have been using and shall continue to use the term "object" interchangeably to refer to the "object abstraction interface." The object abstraction interface exposes a set of capabilities and policies associated with the object. These capabilities are implemented over the set of resources allocated to the object through realm specific mechanisms. Applications need to lease objects belonging to a single or multiple realms to compose their specific application context. Object composition entails "policy negotiations." Realm managers are responsible for management of objects within their realm. Apart from this, the realm managers in each tier participate to implement intra-tier and inter-tier (or cross-tier) management and control functions such object advertisement and leasing framework, monitoring and auditing framework, etc.

The *three tier object* model represents the "policy-plane" of the Internet 3.0 architecture. It exposes the interface over which the "functional-plane" of the architecture is implemented. The "*object abstraction*" serves as the common glue between the policy plane and functional plane by allowing functional requirements of distributed application contexts to be mapped onto object capabilities of objects leased from multiple ownership realms and composed over the policy negotiation and enforcement primitives of the policy plane. In the next section, we shall briefly discuss the "functional plane" of Internet 3.0, followed by a set of object composition principles that are dictated by both the functional as well as policy primitives of the architecture.

4. THE FUNCTIONAL PLANE

The functional plane of Internet 3.0 has three levels of abstraction: **1) Objects***:* Objects represent the basic level of abstraction. **2) End-to-end Services:** End-to-end services represent the next level of abstraction. An end-to-end service is composed from objects of the same type (belonging to the same entity tier) with diverse capabilities. The object's realm publishes an object through a publish/lease framework. Service providers may lease these objects to compose an end-to-end service. Leasing an object entails policy negotiations. The basic and most common policies include, 1) pricing policies laying the terms of usage, and 2) SLA (Service Level Agreement) entailing the capability advertized by the object. **3) Application:** A distributed application represents the highest level of abstraction. Applications specify a "requirement specific context" through a **Requirement Specification (RSpec)** document that is translated and rendered into a **Map** document. The map guides object compositions to specific services and service compositions to distributed applications.

4.1 Objects: Functional Taxonomy

We already defined "objects" in Section 3.4. In this section we present a more detailed functional taxonomy of objects with reference to the entity type that they belong to.

A. Data Objects: *"Data/Content"* is the resource owned by data owners or data realms. *Data* is leased as *Data objects.* A *Data object* is data encapsulated within meta-data. *Capability* pertaining to a data object is represented through its content, availability, number of instances, etc. *Policies* represent the usage terms, security policies, authorization and authentication parameters etc. As an example, a movie 'X' created by the production company 'XYZ Corporation' may be encapsulated within meta-data specifying copyright policies, terms and policies of usage etc and leased to a Video-on-Demand (VoD) provider for distribution. Video-on-Demand is a service provided by the VoD provider by leasing movie objects from multiple production houses. This service is instantiated through a distributed "requirement specific" content distribution application with specific processing, storage, buffering, networking, etc. requirements.

B. Host Objects: Host objects shall typically abstract capabilities of data processing, data storage and other data handling capabilities as required by data services, that specify the context of data handling for a specific distributed application over the typical host resources of processors, memory, caches, secondary storage etc. These resources may belong to separate owners or host realms such as grid computing platforms, cloud computing platforms, individual users, organizations etc.

C. Infrastructure Objects: Autonomous systems (AS) of the present Internet shall be treated as infrastructure realms. Infrastructure realms shall advertize "transit objects". Each transit object shall be characterized by an ingress and egress AS neighbor and transit characteristics in terms of throughput, delay, etc. Internally, a transit object needs to be instantiated as required resource reservations and forwarding table entries to honor the capabilities advertized by the object. It must be realized that objects are expected to represent a high degree of diversity and cannot be classified under broad service classes as in differentiated services architectures. Also, these objects represent

leases that are dynamically instantiated and have temporal validity.

All "objects" can be classified as "real" objects and "placeholder" objects.

D. Real Object: An object that is instantiated physically through an object lease and is embedded as a functional component of an application is called a real object. Unless otherwise mentioned the term "object" always refers to a "real object."

E: Placeholder Objects: A "placeholder object" is a non-instantiated object to which no resources has yet been allocated. A placeholder object is primarily an advertized object that is yet to be leased by an application. There is a one-to-one mapping between a placeholder object and a real object. The difference between the two being that the real object is implicitly bound by a Service Level Agreement (SLA) on its advertized capabilities, while no such SLA binds the placeholder object.

4.2 Object Composition: Principles

The basic premise of the Internet 3.0 functional plane is to allow applications to dynamically compose their specific tiered contexts through composing objects leased over the "Policy Plane." In this section we present the underlying principles that govern object compositions.

Principle #1: Objects represent singular capabilities. These singular capabilities can only satisfy "local requirement specifications" in the map. Thus, "end-to-end requirement specifications" of a map has to be factored into singular object capabilities that can be satisfied by object composition. Such composition shall spawn new end-to-end requirements and re-definition of existing end-to-end requirements.

Principle #2: Derived from Principle #1, if re-defining an end-to-end requirement owing to higher priority end-to-end requirement initiated refactoring, causes the original end-to-end service to be rendered impossible, then the map is considered invalid.

Principle #3: ***Strict top-down ordering of object composition:*** The object composition is guided by the top-down movement of the map and hence enforces a strict top-down order in object composition.

Principle #4: ***Horizontal Composite:*** Simple/Composite objects belonging to the same entity level may be composed together to form a horizontal composite object, if, 1) each object satisfies at-least one singular requirement specification of the map through its set of capabilities, 2) their realm policies can be negotiated, and 3) their object policies for horizontal composite formation can be negotiated.

Principle #5: ***Vertical Composite***: A vertical composite is formed by stacking a simple/composite object of same or lower entity level below another simple/composite object if, 1) the difference in their entity level is at-most one, 2) their realm policies can be negotiated, and 3) their object policies for vertical composite formation can be negotiated.

Principle #6: ***Termination condition:*** The termination of a composition is indicated by the prune function on a map returning an empty set for both local and end-to-end requirement specifications.

4.3 Object Composition: Map Primitive

The basic primitive that drives the mechanism of object composition is called the "MAP primitive." It is characterized by the "MAP" document and a set of functions. The "MAP primitive" lays down a broad framework for object composition. The specific details of the map rendering process are implementation specific.

4.3.1. MAP: A map may be considered to be a type of dynamic "workflow" [22][23][4][11][26]. It represents requirement abstractions that drive object composition. It is a set of requirement specifications defining a particular "requirement specific" networking context. The map presents different levels of abstraction, with different sets of parameters at each entity level and moves top-down through the different entity levels.

The requirements are specified as "local requirements" and "end-to-end (e2e) requirements". A local requirement relates to parameters that can be satisfied by individual objects while end-to-end requirements are spawned when the individual objects are composed into groups. The map initially starts off with a few local and end-to-end requirement parameters defined over placeholder objects (Section 4.1 *Taxonomy* E) at the application specification level. This highly abstract specification mostly provides a top level description of the desired networking context. The context is refined and the abstract service parameters instantiated with actual object capabilities as the map moves downwards.

The key idea is to map the top level context specific requirements into discrete individual object capabilities. The requirements are prioritized at each level. Thus, at each step of the map's descent, it initiates a horizontal composition of objects that satisfy a subset of the map's local requirements. The composition also spawns a new set of end-to-end requirements. The end-to-end requirements of objects belonging to the same entity level are recursively satisfied by factoring an end-to-end requirement as local requirements and initiating a horizontal composition (Section 4.2 *Principle #4*). The horizontal composites are stacked downwards along a descending priority order. Finally, when all the local and end-to-end requirements are satisfied, the composite represents the requirement specific context that was set to be defined.

4.3.2. MAP functions: The three basic functions that implement the MAP primitive are as follows:

1. *Translate:* Replaces the ***placeholder*** objects in local and end-to-end requirements with ***real*** objects.

2. *Prune:* Prunes a translated map to get rid of the local requirements that have been satisfied by the object compositions in that level.

3. *Remap:* draws a new map from translated and pruned map. The group of end-to-end requirement specifications of the pruned map, having the highest priority is remapped to spawn new local and end-to-end requirement specifications in terms of objects that satisfy the end-to-end service required for these objects. The end-to-end requirements not part of this group are redefined in terms of this remapping.

4.4 The Service Level

Realms own and manage objects. These objects are logical abstractions of realm specific services, capabilities and policies. End-to-end services need to be composed by aggregating objects leased from object owners that match the requirements of the specific end-to-end service context. In the object abstraction

concept, end-to-end services are represented as composite objects and can be generally typified as a "horizontal composite" object (Section 4.2 *Principle* #4). Objects are logical entities and implicitly carry an SLA on the advertized capabilities. However, these SLAs are activated only through object lease and instantiation as an integral part of an object composition. Thus, service level SLAs are mapped to object level SLAs. Nonetheless, the executor of the SLA is different in the two cases and so are the stakes on non-compliance.

4.5 The Application Level

Applications are vertical compositions (Section 4.2 *Principle #5*) of end-to-end services. Composing a "requirement specific" application can be abstracted as translating the "RSpec document" to the map document"(Section 4.3) and managing the top-down movement of the "map", through cycles of *translate, prune and remap* functions, as discussed in Section 4.3.2. The design of the management plane at this level presents two distinct choices depending on where the "remap" (Section 4.3.2, Part 3) function is performed. One choice is for the application to perform the remap function, thus requiring the application to specifically spawn each service, and also be responsible for re-composition of the context when required. The other option is for the remapping function to be performed at the service layers with each service spawning the next service layer in a top-down fashion, and thus locally handle re-compositions at and below the service layer that needs to be recomposed. However, at this point, we believe that the architecture shall need to define separate suite of protocols to allow existence of both mechanisms. The first option of having applications in full control of spawning each required service is necessary for realizing the basic goal of top-down diversity wherein the application requirement may be directly instantiated by the application itself in the absence of a service provider providing the relevant service. The second option of delegating the "remap" responsibility to services, allows composite services to be pre-configured, thus defining common contexts in terms of pre-composed services.

5. THE APPLICATION CONTEXT

In this section, we shall discuss a prototype object composition scenario that iterates through the "MAP" functions to dynamically spawn a multi-tier application context starting from an abstract set of requirement specification.

Initially to start with, single and group requirements are specified in terms of application level place-holder objects belonging to the placeholder object space. At the first level of object composition, when the place-holder object's local requirements are instantiated over actual objects belonging to the object space, the map is translated. After this the map is pruned and then remapped to guide the next level of object composition. Thus, the process of object composition involves multiple cycles of *(translate, prune, remap)* of the original map till 'prune' returns a map with empty local and end-to-end requirement specification sets.

Figure 4, is an example of the map rendering procedure along the different service levels, discussed thus far. The highest level application layer represents requirements between a data source and the data sink. An intermediate set of data processing objects processes the data produced by the data source. Between the first two data processing host objects, a delay tolerant service needs to be interposed that provides capability to store the data till a forwarding link is available. Finally all these objects map to infrastructure objects for actual transmission. The interposed host entity level services introduce packet transmission delays between the data-source and data-sink, thus requiring the "transit objects" over the infrastructure to vary in their capabilities. Also, as shown in the figure, at each downward step, the local requirements are instantiated on real objects and the end-to-end requirements are rendered into multiple placeholder objects.

An Example Application Scenario: The scientific community is generally far spread out geographically and so are scientific installations. Experimenters generally need to run experiments remotely on huge shared experimental installations. These experiments often produce enormous amounts of data that might need to be fed to another experiment that depends on this data. Also, huge amount of data need to be stored and made available for processing or analysis when required. An experimental setup consisting of geographically spread out scientists, scientific installations producing/consuming data, supercomputers churning humongous amounts of data require the services of a massive and extremely efficient distributed system. The resource requirements in terms of data processing and storage, and real-time bounds on delay of peta-scale data transfer, force scientific experiments to be conducted on specially built grid computing facilities that are served by dedicated networking resources [6][3][10]. Such specialized facilities incur huge setup costs and time that need to be amortized through sufficient long term usage. The basic design objective of the multi-tiered diversified architecture of Internet 3.0 is to enable the Internet to serve as the substrate of such large scale requirement specific applications. Being able to utilize the distributed shared resource base of the Internet with the required degree of isolation and QoS, provisioned for the duration of the scientific experiment has huge gains in terms of cost savings, low barrier to entry, high utilization, high availability and increased robustness owing to higher redundancy.

In the context of multi-tier diversification, an experimental setup run by geographically spread out data objects producing/consuming huge amounts of data can be composed as a high level distributed service. Distributed data processing objects and storage objects leased from multiple host facilities such as supercomputing centers, cloud computing platforms, etc., can be composed into an end-to-end host service by end-to-end host level service providers. These processing and storage objects together with the data source and data sinks need to be connected by infrastructure objects with specific delay and throughput properties. Infrastructure objects satisfying these connectivity, delay and throughput requirements can be leased from multiple infrastructure object owners and composed into end-to-end paths by infrastructure level service providers. Thus, the specific requirements of distributed scientific experiment contexts can be served over the shared Internet facility through object abstraction and object composition in a multi-tiered diversified networking architecture.

Figure 5 represents a policy view of the above scenario. Each tier spawns a virtual cloud representing the entities of that tier. The virtual cloud represents a common policy horizontal composite of objects from that tier. The functional requirement of the application context is satisfied through the recursively calling the *translate, prune* and *remap* function on the application level specification. The dotted line between each tier represents a service layer through which specific tier-based capabilities are advertized and leased as horizontal composites. The application itself is a vertical composite of tiered service layers.

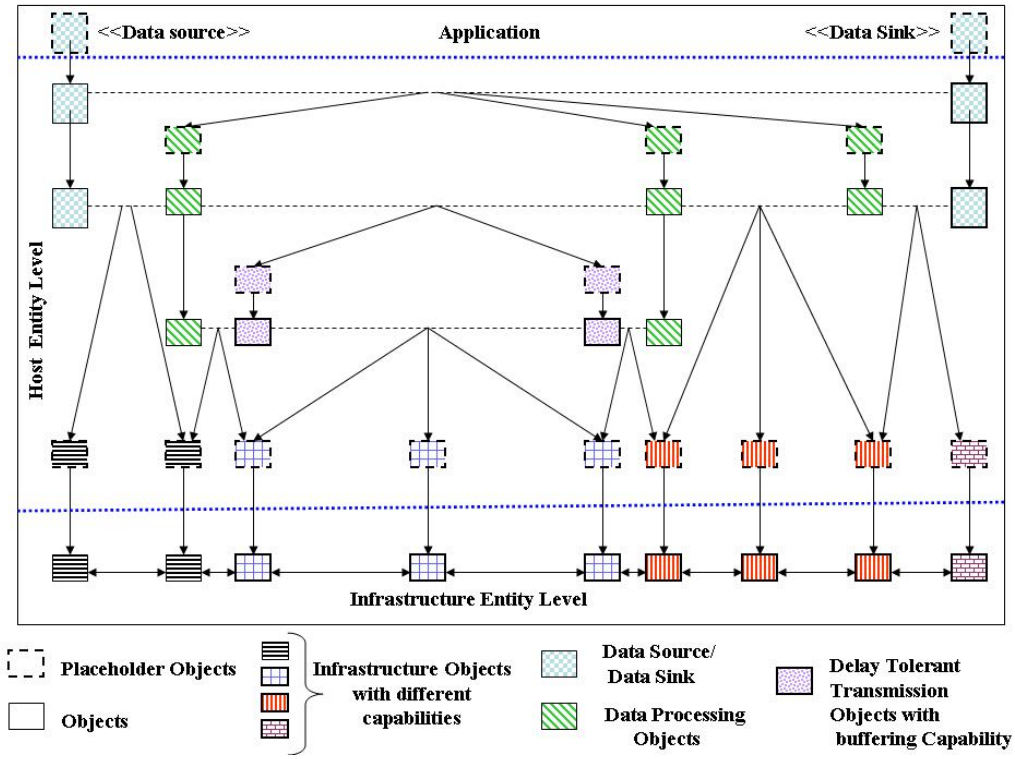

Figure 4. Example Scenario of "Rendering the Map"

The exact mechanisms for the map rendering process is implementation specific. Several different map rendering mechanisms could co-exist over the basic primitive of object abstraction. The policy framework and the functional plane provide clean interfaces for management, control and policy negotiation that allow the process of spawning a whole application context across resources leased from multiple ownerships to be completely automated. This allows novel application contexts to be defined and deployed on-the fly. Also, it allows applications to manage their commercial viability better by allowing all resource lease/allocation and de-allocation to be dynamic over a multi-ownership framework.

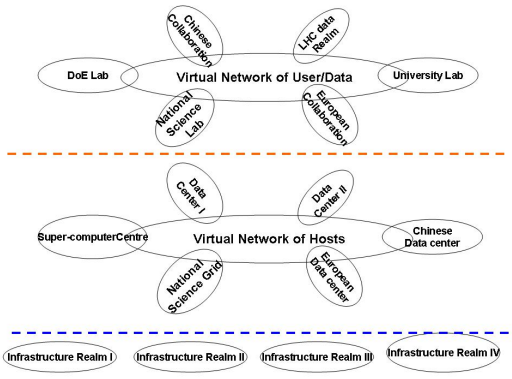

Figure 5. Example Scenario of "Rendering the Map

Some of the features of application contexts over the Internet 3.0 framework are:

1. Dynamically Configurable: A distributed application context may be spawned dynamically over the distributed resource base of multiple owners. Also, the framework allows applications to dynamically re-configure (scale up/scale down/ release) its context allowing it to be suitably manage its performance and be economically feasible.

2. QoS Mapping and Abstractions: "Requirement specific networking" is driven by the need to represent QoS requirements of individual contexts and mapping it to distributed resource capability parameters that represent specific requirements from the networking substrate. Multi-tier diversification through the "object abstraction" is a direct manifestation of multi-level QoS specification and mapping [17][18][14][15][5][12]. Thus, an application QoS specification is mapped to a distributed service QoS parameters. The service QoS parameters are further mapped into host parameters including number of hosts with the service instance required, location of such hosts, host capabilities such as connectivity, storage, memory, processing power, etc, and finally need to be further mapped into infrastructure path parameters such as delays, throughputs, link speeds etc. There may be many more intermediate levels of mapping and at each level the requirement for that level may be further mapped to specific device parameters such as buffer sizes, queuing delays, memory cache levels, processor parallelism etc. The "tiers" in the multi-tier diversification architecture represent levels of object abstractions in this multi-level mapping and present interfaces exposing capabilities, policies, etc. in some standard representation.

3. Fine Grained Policy Enforcements: The current Internet design conflates identities, obscuring ownerships. Separate ownerships of infrastructure, hosts, and data cannot be explicitly represented in the current architecture. Lack of explicit representation of ownership makes it extremely difficult to enforce policies at the required level of granularity. This creates tussles. Examples of such tussles are abundant in the form of deep packet inspections, port blocking, IP blocking and other such mechanisms by data path proxies to enforce organizational policies on incoming/ outgoing data and organizational host access, ISP's throttling P2P overlay traffic that do not conform to ISP routing policies, P2P traffic avoiding ISP rationing through end-to-end encryptions, etc. The future diversified Internet need to explicitly establish the ownership of objects and foster an environment of active negotiation such that owners of objects can specify, negotiate and enforce their policies at the required level of granularity and object leases implicitly entail a service level agreement between object owner towards the leaser.

4. Co-operative Business Incentives Aligned to Promote Diversified Choices: The "object abstraction" in Internet 3.0 de-conflates functionality from policies by separating end-to-end services provided by service providers from object owners. This separation creates a unique business environment of cooperative competition wherein object owners co-operate with each other to make their objects more attractive to end-to-end service contexts and at the same time compete to make their objects better than others to guarantee their lease amongst similar existing choices.

5. Well Defined Accountability: One of the biggest hurdles in enforcing service level agreements (SLA) [1][20][25][19][24] into the current Internet is the lack of architectural support for defining a framework to ascertain accountability. The Internet 3.0 architecture is proposed to be designed with SLA enforcement as a basic requirement. "Objects" will be the building blocks for any network context. These objects will advertize capabilities. A leasing framework will allow leasing these objects to services. This lease will implicitly enforce an SLA on the advertised capabilities of the object, between object owners and the object lessee. Additionally, the management plane of Internet 3.0 will provide inherent architectural support for a multi level monitoring and measurement framework allowing monitoring and measurements at different levels of aggregation.

6. Flexibility and Future Ossification: The primary reason for the current ossification of the Internet can be attributed to 1) huge investments in the current technology, 2) need for extensive and concurrent multi-lateral changes, and 3) lack of enough business incentives. The design of multi-tier diversification will try to address these hurdles through, 1) "object abstraction" encapsulating diversity and exposing a standardized abstract interface, 2) allowing current stakeholders to trade their capabilities and resources as objects, 3) creating newer business opportunities in the form of diversified service providers creating newer and disruptive services from objects leased by object owners, and 4) allowing each party to express, negotiate and enforce their individual policies. The Internet 3.0 architecture is designed to create an architectural framework wherein diverse technologies and associated protocols can co-exist under a uniform management and control framework and co-operate to define requirement specific networking contexts, thus providing huge flexibility gains and preventing future ossification.

6. SUMMARY

In this paper, we propose a framework wherein the availability of enormous amount of compute resources may be put into the correct perspective and evolve an integrated "communication paradigm" based architecture over the present "communication-system" based architecture. The communication-paradigm based architecture shall enable novel distributed applications to be dynamically deployed and managed over resources leased from multiple ownerships. Also, the three tier perspective of the communication-paradigm based architecture over a recursive application integration platform allows an application context to be highly "verticlized." The salient feature of the architecture is that it overlays a policy framework on the functional plane, thus allowing the functional plane to evolve more naturally and legitimately over the multi-ownership nature of resources serving the application context.

7. REFERENCES

[1] AT&T Managed Internet Service (MIS), http://new.serviceguide.att.com/mis.htm, 2007.

[2] R. Braden, D. Clark, S. Shenker, **"**Integrated Services in the Internet Architecture: an Overview," RFC 1633, June 1994

[3] CERNET, www.edu.cn/cernet_1377/index.shtml

[4] Marco Danelutto; Paraskevi Fragopoulou, Vladimir Getov (Eds.), "Making Grids Work," Proceedings of the CoreGRID Workshop on Programming Models Grid and P2P System Architecture Grid Systems, Tools and Environments, Heraklion, Crete, GreeceSpringer US, 12-13 June 2007, pp 309-321

[5] L. A. DaSilva, "QoS mapping along the protocol stack: discussion and preliminary results," IEEE International Conference on Communications, 2000, Volume 2, pp 713-717

[6] ESNet, http://www.es.net/

[7] Joan Feigenbaum, Rahul Sami, Scott Shenker, "Mechanism design for policy routing," Journal of Distributed Computing, Springer, Volume 18, Number 4 / March, 2006, pp 293-305

[8] Nick Feamster, Ramesh Johari, Hari Balakrishnan, "Implications of autonomy for the expressiveness of policy routing," ACM SIGCOMM Computer Communication Review, Volume 35, Issue 4 (October 2005), pp 25 - 36

[9] L. Gao, "On Inferring Automonous System Relationships in the Internet," IEEE Globe Internet, Nov 2000

[10] GEANT, http://www.geant.net/

[11] A. Hoheisel, "Grid Workflow Execution Service–Dynamic and interactive execution and visualization of distributed workflows," In Proceedings of the Cracow Grid Workshop 2006, Cracow, 2006.

[12] Jean-Francois Huard, Aurel A. Lazar, "On QOS Mapping in Multimedia Networks," *Computer Software and Applications Conference, Annual International*, pp. 312, COMPSAC '97 - 21st International Computer Software and Applications Conference, 1997.

[13] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking Named Content. In CoNext, Rome, Italy, December, 2009

[14]Jingwen Jin, Klara Nahrstedt, "QoS Specification Languages for Distributed Multimedia Applications: A Survey and Taxonomy," IEEE MultiMedia, vol. 11, no. 3, pp. 74-87, July 2004

[15] Klara Nahrstedt, Duangdao Wichadakul, Dongyan Xu, "Distributed QoS Compilation and Run time Instantiation," Proceedings of IEEE/IFIP International Workshop on QoS, 2000

[16] K. Nichols, S. Blake, F. Baker, D. Black, **"**Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474, December 1998

[17] C. Koliver, K. Nahrstedt, J. Farines, J. D. Fraga, and S.A. Sandri, „Specification, Mapping and Control for QoS Adaptation," *Real-Time Syst.* 23, 1/2 (Jul. 2002), 143-174.

[18] B. Li and K. Nahrstedt, "A control-based middleware framework for quality of service adaptation," *IEEE Journal on Selected Areas in Communications (JSAC)* 17(9): 1632-1650

[19] J. Martin and A. Nilsson, "On service level agreements for IP networks," In *IEEE INFOCOM '02*, 2002.

[20] NTT Communications, "Global IP Network Service Level Agreement (SLA)," http://www.us.ntt.net/support/sla/network/, 2007

[21] Y. Rekhter, T. Li, S.Hares, Editors, A Border Gateway Protocol 4 (BGP-4), RFC4271, Januay, 2006

[22] J. Wang, D. Rosca, W. Tepfenhart, A. Milewski, and M. Stoute, "An intuitive formal approach to dynamic workflow modeling and analysis," In *Proceedings of the Third International Conference on Business Process Management* (Nancy, France, 2005).

[23] B. Shafiq, A. Samuel, H. Ghafoor, "A GTRBAC based system for dynamic workflow composition and management Object-Oriented Real-Time Distributed Computing," 2005, ISORC, 18-20 May 2005 Page(s): 284 – 290

[24] A. Shaikh and A. Greenberg, "Operations and Management of IP Networks: What Researchers Should Know," Tutorial Session, ACM SIGCOMM '05, August, 2005

[25] Sprint NEXTEL, "Service Level Agreements," http://www.sprint.com/business/support/serviceLevelAgreements.jsp, 2007.

[26] J. Yu, R. Buyya, "A Taxonomy of Scientific Workflow Systems for Grid Computing," SIGMOD RECORD, VOL 34; NUMB 3, 2005, pages 44-49