Thesis Proposal

Han Gao

hangao@cs.uchicago.edu

The University of Chicago

Computer Science Department

1100 East 58th Street

Chicago, Illinois 60637

Technical Report DRAFT

March 20 2002

Abstract

Based on the concepts of computational grid and future laboratory, we have a broader view for computer science development. Network service as one kind of computational resources, will be integrated and applied into more and more network related applications. This paper describes a multi-layer framework model for utilizing and developing network services and propose some new features for future architecture design. We also exemplify this theoretical model to an useful grid application, Access Grid.

The University of Chicago Computer Science Department and Argonne National Laboratory Mathematics and Computer Science Division supported this work.

1 Introduction

Accelerated by the technologies of high speed networks, inspired by the concepts of Grid (5; 4) and Future Laboratory (2; 3), an advanced software architecture is required for ubiquitous access to the network resource (6). The heterogeneous network characteristics, various device capabilities and specified requirements always make the application have unacceptably poor performance and furthermore, the network resource have unexpected low utilization. Network service, as an emergence of new network resource, become one efficient solution for those situations. Consider the following scenario as example (Figure 1). Alice, a researcher in London, wants to use Access Grid to collaborate with her research-mates in Chicago. However, in some reason, the media device in Chicago can only accept 16kHz audio while Alice can only produce 8KHz audio stream. After capability negotiation in the AG venue, the network service engine is invoked. It discovers the appropriate transcoding engine, which transforms the 8kHz audio stream to 16kHz. Then, it reconfigures the route of the lower resolution data to this network service and direct the output data to the media device in Chicago. During that procedure, the specified modules such as broker, monitor, management and other components in the framework will be introduced, while it is transparent and unaware for both users in Chicago and London.

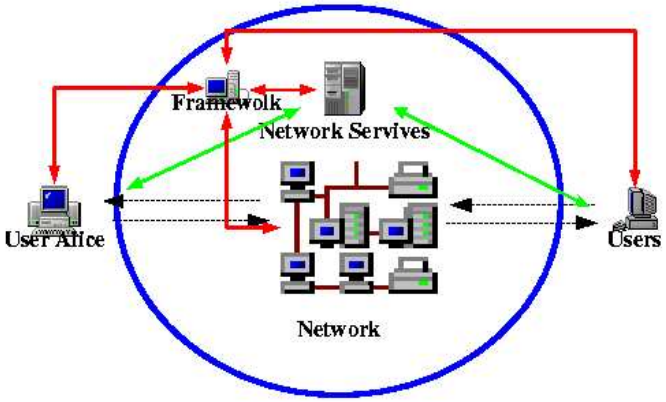

Figure 1: **AG Scenario**: the black lines and arrows composite the view for application users; the green tracks composite is actual data stream route; the red track composite the running plan of network service framework.

However, the current network services remain many open issues to be resolved. The higher demands and

faster development of scalable services encourage us to design a framework such as an engine to drive them running smoothly. In this paper, we present a theoretical model for network service engine - multi-layered software architecture. This framework should include the following nice properties for the current and future requirements:

- Transparent: the running procedure of network service engine is as transparent as possible to each end users. The application server invokes the network service engine and the afterward steps are processed by our framework.
- Self-adaptable: the framework can provide a global adaptation during network service running. Reconfigure the necessary system components to handle the dynamic and non-local changes.
- Reusable: The framework can provide a general model for different and complex end application. According various characteristics and sophisticated requirements, each application can adjust the framework to the specified goals. The layers and components of framework can be plugged in and out by different designs.

2 Theoretical Model

We views the future network as composited by resource objects, user applications and links connect the former two components. We summary some new definitions and properties for the network development:

- Resource : the definition of network resource have more broader and general meaning: it includes data objects, data streaming, network application, network service, etc.
- Stream: according the high transfer rate of networks and high processing rate of processors, we can consider all the resource objects as streaming with various format. The framework have ability to deal with large scale data streaming.
- Soft-state: each layer or component will exist and be maintained via a mixture of explicit and probabilistic means. For specified purpose and unexpected non-local changes, the framework have ability to reconfigure it self to satisfy the requirements.

We propose a general framework model as guidance or reference for the development and utilization of network services engine. Besides the nice properties and new concepts we discussed above, our multi-layered architecture (Figure 2) should be viewed as following:

- Application layer: users can plug their various resources, applications and services from different domains in this layer. For example, there are several service domains from grid applications: bioinformatics, weather, aerospace , etc.
- Management layer: the matcher module and interceptor module are centerpieces in this layer. Interceptor can translates the application requirements, system capabilities and network configuration to the specified parameters which our framework can understand and process. Matcher negotiates with sets of requests and capabilities and return the best solutions. Also, it can translate the processed or result parameters back to application layer. We can also plug monitor, discovery modules in this layer.
- Stream layer: stream routing and processing are the main tasks in this layer. This layer play a main role when the system need reconfiguration or self-adaption due to the global dynamical change.
- Transport layer: this layer acts as transportation and topology policy maker. It decides which kind of transportation and topology we should deploy.

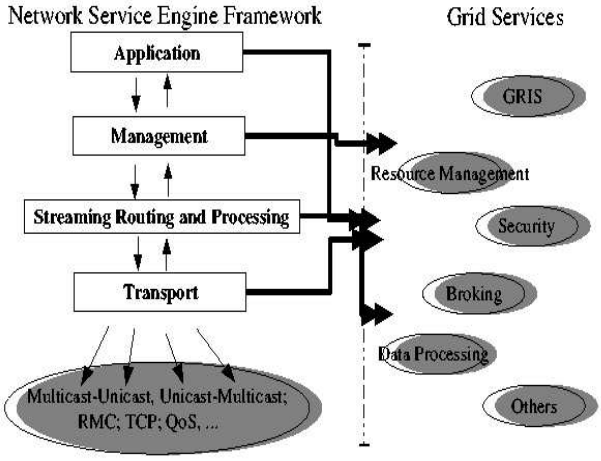

Figure 2: **General Architecture**:

3 Current Work and Strategy

We will implement the general architecture into AG network service engine as an example.

In the first three months, we will discuss and re-design the theoretical model for the network service framework. The first goal is that the framework can execute the basic functionality properly. We will implement the interceptor module, capability negotiation module, into AG 2.0. So, in this stage AG can run the basic and simple network service.

From late autumn to late spring of next year, we want to implement the basic self-adaption to this framework as a second goal. This plan will include implementing the routing and topology makers (some polices such as multicast to unicast or vice versa, QoS, TCP), stream routing and processing, self-reconfiguration for global changes.

The third goal is integrating each layer, component and module as our framework and make enough experiments. Also, we want to implement and improve this framework by OGSA standard (7). This will last from summer to spring of the coming year.

In each end of stage, we will publish the general ideas, important technical details and result summaries.

4 Technical Details

Capability matching and negotiation modules will be implemented as core for network services engine. The following discussion will give more algorithms and technical details for AG 2.0 network service engine implementation.

4.1 Interceptor Modules

We use XML as standard format to describe requirements and capabilities of service, applications and systems. It translates the specific parameters to the script which matcher can understand. Due to the flexible and various characteristics of each description file, we embed the ClassAd (9) into XML file. This

semi-structure description language can explicitly describe any requirements, configurations and solutions. We use corresponded ClassAd library to evaluate any attributes or expressions we will use in the matcher module.

4.2 Matcher Modules

Our goal is finding the best resolution for all the client nodes. For example, for audio capability resolution, we should find the best resolution which all clients can encoding and decoding, so they can interact each other. For the following algorithms, we just use audio capability negotiation as an example.

The first algorithm is matching two audio capability description files, which finds either the preference resolution or best resolution both two clients can afford (Alogrithm 1).

Algorithm 1 Matching capabilities for two nodes

if type of A is compatible with type of B **then**
  if one preference for both nodes **then**
    return the preference
  else
    $L \Leftarrow$ all the possible codes for both nodes
    Initialize the resolution list $R = NULL$
    for each element E in the list L **do**
      if find a best resolution r for encoding method E **then**
        $R \Leftarrow append(R, r)$
      end if
    end for
  end if
else
  return "no possible matched resolution"
end if

However, the situation always not be so simple for only two client nodes. We should match one node with one set which already have a contract among their n nodes for interaction. After add this one node into

-	L16	L8	PCMU	PCMA	GSM	...
48K-Stereo	$n-4$	$n-2$	10	11	4	...
48K-Mono	n	$n-1$	$n-5$	2	8	...
32K-Stereo	$n-2$	9	11	$n-1$	n	...
...						

Table 1:

-	L16-16K-Mono	L16-8K-Mono	L8-16K-Stereo	...
L16-16K-Mono	-	7	$n-2$	...
L16-8K-Mono	7	-	4	...
...				

Table 2:

group, we need re-negotiate the capabilities for the new $n+1$ nodes set. Same with the situation when we disjoin one node from the groups and rest of nodes become a new $n-1$ nodes set. In both add-in or out, the algorithm should handle the dynamic change in the set. We introduce a capability table here, each cell will record how many nodes have the corresponding resolution.

From the above table, we can obtain the best resolution in which cell the value is n (Algorithm 2). For n nodes, the complexity of this algorithm is only $O(n)$.

Besides the client nodes, we will add our network service into our group. As the very beginning expample, we need a audio transcoding network service. One hand this make the group can comunicate with each other via different audio resolutions, in the other hand, it also make things more complex. Here we introduce service table (Table 2).

This table records each serivce, and also in each cell records how many nodes do not need this service. Suppose in the n nodes set, we add one network service in which it can translate L16-16K-Mono to L16-8K-Mono and vice versa.

4.3 Self-adaption

Algorithm 2 Matching capabilities between one node with a *n* nodes set

```
Input capability of node x
for each resolution r of x do
  if r not in the table then
    add a new resolution entry to the table
  else
    change table to M*N
    add 1 to the corresponding cell value
  end if
end for
for i = 1 TO M do
  for j = 1 TO N do
    if [i,j]=n+1 then
      return the resolution
    end if
  end for
end for
```

Self-adaption will be another difficulty in our system. The exception will be caused by either hardware failures or dynamic non-local changes during running times. Some (8; 1) give us some general ideas how to deal with them.

References

[1] S.-W. Cheng, D. Garlan, B. Schmerl, P. Steenkiste, and N. Hu. Software architecture-based adaptation for grid computing. *The 11th IEEE Conference on High Performance Distributed Computing (HPDC'02), Edinburgh, Scotland*, July 2002.

[2] L. Childers, T. Disz, M. Hereld, R. Hudson, I. Judson, R. Olson, M. E. Papka, J. Paris, and R. Stevens. ActiveSpaces on the grid: The construction of advanced visualization and interaction environments. pages 64–80, 1998.

Algorithm 3 Matching capabilities between one node with a *n* nodes set

```
Input capability of node x
for each resolution r of x do
  if r not in the table then
    add a new resolution entry to the table
  else
    change table to M*N
    add 1 to the corresponding cell value
  end if
end for
for i = 1 TO M do
  for j = 1 TO N do
    if [i,j]=n+1 then
      return the resolution
    end if
  end for
end for
```


[3] T. Disz, M. E. Papka, and R. Stevens. UbiWorld: An environment integrating virtual reality, supercomputing, and design. pages 46–59.

[4] I. Foster. The grid: A new infrastructure for 21st century science. *Physics Today*, 55(2), February 2002.

[5] I. Foster and C. Kesselam. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, san Francisco, Calif., 1999.

[6] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. Grid services for distributed system integration. *Computer*, 35(6), 2002.

[7] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. *Global Grid Forum, Draft 2*, July 17 2002.

[8] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti. Cans: Composable, adaptive network services infrastructure. *USENIX Symposium on Internet Technologies and Systems*, 2001.

[9] R. Raman, M. Livny, and M. H. Solomon. Matchmaking: Distributed resource management for high throughput computing. In *HPDC*, pages 140–, 1998.