IP Network Configurator: An Infrastructure For IP Network And Services Provisioning

T. Naik[1], G.K.Swanson[1], S. Mazumdar[2], P. Krishnan[3], R. Sequeira[4] and R. Ganesan[5]

[1]Bell Laboratories, 101 Crawfords Cr Rd, Holmdel, NJ 07733, USA
E-Mail : tnaik@research.bell-labs.com, gks@dnrc.bell-labs.com
[2]Avaya Laboratories, 101 Crawfords Cr Rd, Holmdel, NJ 07733, USA
E-Mail : mazum@research.avayalabs.com
[3]Avaya Laboratories, 233 Mt. Airy Road, Basking Ridge, NJ 07920, USA
E-Mail : pk@research.avayalabs.com
[4]ISPSoft Inc, 661 Shrewsbury Avenue, Shrewsbury, NJ 07702, USA
E-Mail : roshan@ispsoft.com
[5]Cosine Communications Inc, 1200 Bridge Parkway, Redwood City, CA 94065,USA
E-Mail : gramu@cosinecom.com

The IP network configurator (IPNC) is an extensible, scalable infrastructure for IP network and services provisioning. It provides necessary components to support high-level management applications. The most important goal of the IPNC is to enable network wide management of the IP networks while encapsulating the details about different network element vendors, different command line interface (CLI)/SNMP versions, different access methods etc. This provides high level APIs to applications shielding most of the element specific details. The IP network configurator tool currently supports CLI and SNMP to manage the IP networks transparently. It analyzes the issues related to configuration of the network of routers to consistently provision associated IP services with a minimal user input. Our architecture is applicable to all IP services and we specifically describe our implementation of the provisioning of the OSPF routing protocol.

Keywords: IPNC, IP Network and Services Provisioning, Service Provisioning Infrastructure.

1. Introduction

1.1. Problem

Dominance of IP networks has created significant need for faster network and service management tools. One of the important problems in IP network and services management is the lack of a scalable provisioning infrastructure that can be extended as new applications and network equipment vendors emerge. These applications can be traditional network management applications or emerging applications in service domain. Significant advances in configuration management domain can help address some of these problems. One of the critical tasks for provisioning of network-based services is configuration management.

One of the biggest challenges in configuration management is that configuration information is replicated at all or several network elements that are part of the network. For example, DNS server information is configured at all the routers. For OSPF, the OSPF area related configuration is replicated over to all the routers in that area. Thus validation and consistency of the configuration information about same entity across multiple network devices becomes a critical task for configuration.

The configuration information of the networks in IP network is stored, in the form of configuration files, in the routers. Since the router acts as a repository for the configuration information, the configuration of the network requires configuration of multiple routers. The configuration management of routers is very much vendor dependent. Some vendors prefer change of configuration information through the router specific command line interface (CLI) or loading of configuration file via tftp. Others prefer SNMP or TL1. Some others also support JAVA RMI, CORBA or HTTP interface for their EMS. As a result there is a no uniform way to configure routers from multiple vendors. The lack of uniform way to configure routers poses a real problem for configuration of network or network services as a whole because any change may require changing configuration of multiple routers from different vendors. Currently most routers support CLI and SNMP protocols for accessing information from routers and configuring the routers. The CLI interfaces are used to modify the configuration file and the messages are sent through a telnet session. The CLI commands are used to configure various components of routers by adding, deleting or modifying the components from the configuration files. The SNMP protocol is primarily used for monitoring and analysis of the network behavior though it can be used for configuration also in principle.

The state of the art in configuration management of router is manual modification of the configuration files of the routers. Although there exits several tools for device level configuration management (e.g. EMS supplied by the NE vendors), there are very few tools that provide on-line interactive provisioning of IP networks and supported services. Part of the reason is that IP networks span over heterogeneous network elements from different vendors and part of the reason is that SNMP protocol is used only for monitoring purposes. Difficulties involved in generic modeling of services exacerbate the problem. Currently, most network administrators log into multiple network elements (routers, switches etc.) and provision the network or services by issuing a set of commands using the element specific command line interface. Unfortunately these commands are vendor specific and as such it is very difficult to build generic tools.

Although we can use SNMP to configure the IP protocols and services, the reality is that many cases the MIBs are not fully implemented (e.g. row creation/deletion is not allowed), SNMP SET is disabled through configuration because of security risk or device level MIBS are not really modeled for network level configuration. The device level standardized MIBs only acts as repository of data but does not really model the IP services that need to be configured. CLI tend to have more configuration capabilities than other management protocols. This makes it very important to support CLI in a management tool. However, proprietary syntax of CLI is a big problem while dealing with a multi-vendor networks.

1.2. Our Solution

From previous discussion it is imperative that there is a need for an infrastructure capable of shielding the heterogeneity of network elements. The heterogeneity could be due to different equipment vendors, different device access protocols e.g. telnet, tftp, SNMP etc. or due to different configuration methods e.g. CLI, SNMP, TL1 etc. It is also necessary to provide high-level APIs through sophisticated network and services models. IP Network configurator (IPNC) provides this infrastructure. It has a sophisticated model of IP Network. Currently IPNC provides APIs for OSPF, RIP and BGP[1]. Almost entire code is written in Java and it provides Java Swing based application as well as applet as front end GUI to IPNC server. Some of the features of IPNC are listed here.

Technical Features:

- Network wide configuration
- Support for multi-vendor network elements
- Transparent access to router using SNMP or CLI over telnet, modem, tftp
- Java based framework for adding support for new protocols
- Configuration sequencing
- Protocol specific logical network views
- Consistency checking of network wide configuration and elimination

[1] RIP and BGP APIs are not fully supported currently.

- Router configuration file version management and disaster recovery
- Topology discovery
- Bulk configuration updates

Section 2.3 gives details about network wide configuration. Section 4 describes how IPNC support for multi-vendor network elements and transparent access. When configuring an IP network, changes may need to be made to several devices, as part of one configuration operation. Given an in-band access to the devices, it becomes imperative to sequence the changes intelligently to retain connectivity to the devices that need to be updated. The IPNC addresses this issue by implementing sequencing algorithms. The IP network configurator currently presents three views of the network: IP network view, OSPF view and Device View. Section 2.3.1 describes IPNC views. As more and more protocol and services are added to the tool, a new view will be added to each protocol/service. The various network views present protocol specific topological view the network in a hierarchical form. The hierarchy for each view is derived from the information structure of protocol or service being managed in that view. We shall not describe other features in more detail in this paper [1].

1.3. Related Work

While many of the OSSs used in telephony world have embraced this idea of network wide provisioning, it is not a normal practice in IP world. The IPNC provides a unique platform for network wide IP provisioning. Several companies are trying to pave their way in service provisioning tools e.g. Astracon[5], Orchestream[8], Syndesis[9], Vertel[10] etc. It is unclear however, how modular, scalable and open some of these platforms are. Netsys [6] from Cisco helps detect inconsistencies in the network. Few companies tried to mimic GUI for CLI but this is not really a network wide provisioning.

1.4. Organization of the Paper

We describe software architecture of IPNC in section 2. Section 2 also compares IPNC architecture to TMN's layered management architecture. Section 3 describes IPNC implementation architecture. Section 4 describes information model of the router agent components. Finally, we present our conclusions in section 5.

2. Software Architecture

The architecture of the IPNC is based on a layered architecture as shown in Figure 1. Although the layering of the IPNC can be viewed as an implementation of the TMN's layered management architecture, in our approach we have adopted multiple messaging protocols instead of one normally prescribed by the TMN. Our approach is also slightly different than the TMN layered approach. Unlike TMN, IPNC layers export API for the higher layer instead of implementing complete applications at these layers. Also, our implementation is tailored for the configuration management, though our architecture can be extended to support other management activities easily.

2.1. Network Element Layer

At the bottom of the IPNC layered architecture is the network element layers that represents the network elements, such as routers, that are to be configured. The entities in the higher layer represent the entities in the network element layer. Current implementation of IPNC only supports the Routers.

2.2. Network Element Model Layer

On top of the network element layer is the element management layer. The element management layer consists of three sub-layers: device access protocol layer, device access protocol mediation Layer and device model layer.

2.2.1. Device Access Protocol Layer

The device access protocol layer provides an abstraction of multiple protocols that can be used to communicate with the network elements. A particular network element can be reached through different access protocols depending on its capabilities. We assume that most of the network elements can be reached through telnet and some by modem and/or SNMP. The device access layer also provides access to specific network elements through indirect telnet – a telnet session through one of the neighboring network elements instead of direct session from management station. Indirect telnet can be very powerful technique especially when one needs to connect to a router to fix the very problem that prevents one to connect to that router at the first place.

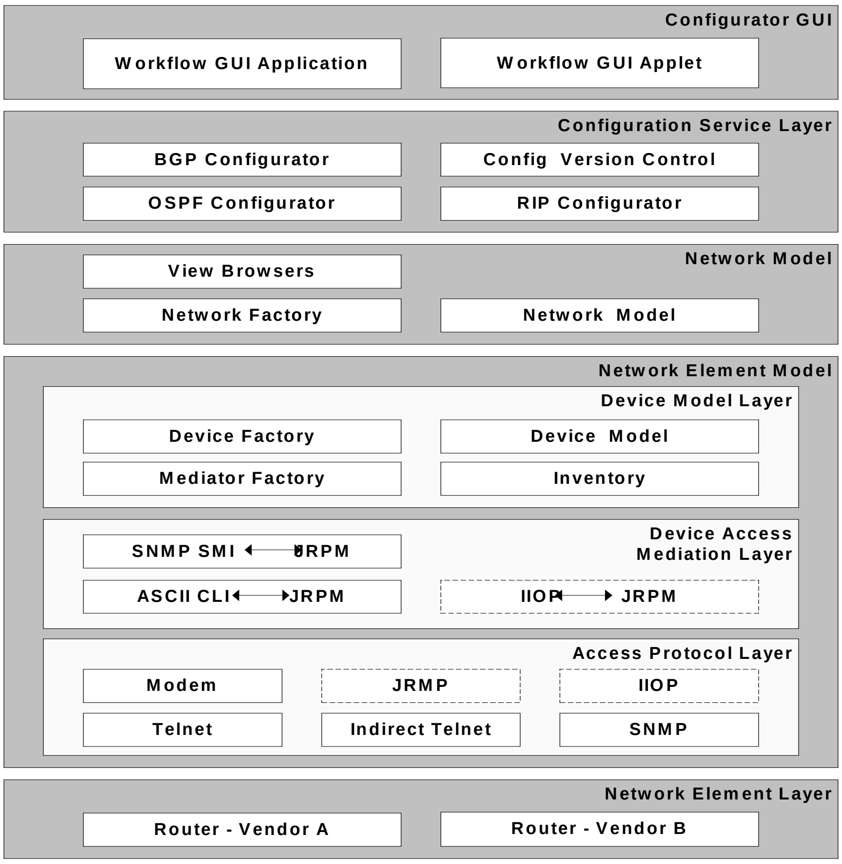


Figure 1. Layered Architecture of the IP Network Configurator

2.2.2. Device Access Mediation Layer

Since the device access protocol layer supports multiple protocols to communicate with the network elements, we need mediation service that can translate an implementation specific representation of the network information to the multiple message formats supported by device access protocol layer. The device access mediation layer provides a service for mapping information in the device model layer to the specific message protocol in the device access layer. In IPNC this layer provides vendor neutral Java RMI based APIs to higher layer and converts generic requests to device access specific format internally. This involves support for different vendor CLIs, SNMP etc.

2.2.3. Device Model Layer

The Device Model layer represents the partial software image of the managed networks elements within the scope of the IPNC tool. The entities in the device model layer correspond to the entities of the network element layer. The entities in the device model layer not only represent the hardware elements but also the software protocol entities that are also running in those hardware elements. Specifically for our case, device layer entities represent the routers, its link interfaces, configured IP interfaces on Link interfaces, software protocol entities that represents various IP services, such as routing protocols (OSPF, BGP, RIP etc.), management protocols (SNMP) and others. The entities in the device model layer represents the information needed to configure the network elements in network wide way. Currently, IPNC supports configuration and status. The IPNC can be easily extended to support statistics.

The device model layer also provides a set of services: Inventory, Device factory and Mediator factory. We use the inventory service to store the name, access information and some additional information about the routers within the scope of the IPNC tool as a persistent storage in a file. The inventory service also interacts with the element discovery tool to discover routers with the scope of the IPNC tool. The factory services provide mechanism for creating the device entities in a distributed way. The mediator factory provides mechanism to find vendor and device dependent mediation and access protocol entities for a specific device. The access to the mediator factory is only available to device factory. The device model layer abstracts the differences in the heterogeneous routers from multiple vendors and provides vendor, version and access independent model to the network layer.

2.3. Network Model Layer

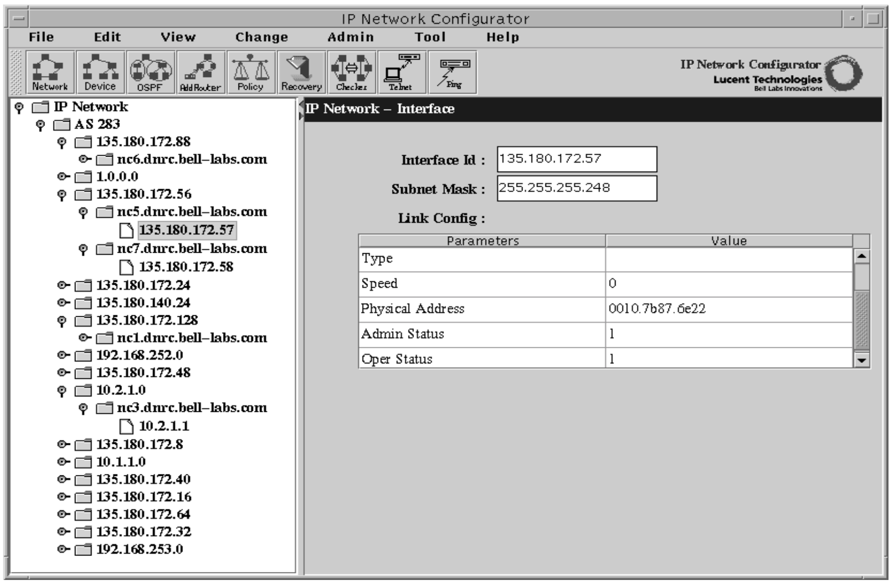
The network Model layer maintains the relationship between the network elements, i.e. the routers (for our case), in a set of the network entities. The network entities are abstract entity that represents the connectivity between the network elements and also the hierarchy of the network elements. Network model maintain multiple graph (relationships between network entities), one for each protocol or service the routers support. For example, the network model maintains one graph for all the routers that are running IP protocol, one graph for all the routers that are running only the OSPF protocols. Based on the protocol specific information, we try to build hierarchy of the network. For example, in case of IP we maintain the hierarchy of the autonomous system (AS) and the IP subnets. The IPNC tool, the network is configured by changing the relationship between the hierarchy of network, the relationship between the routers and the network level parameters. The network elements that are part of the network get automatically configured as a part of the network configuration. Although we support configuration of devices explicitly, we do not recommend it because there is potential that the device configuration may not be consistent with the network wide configuration. For example setting OSPF hello interval on an interface to a value that is different form other interfaces on the same subnet will result in communication failure at OSPF layer. This can be avoided if that value is changed from OSPF subnet view, which results in setting the same value in all interfaces in that subnet. The network elements are configured based on specific protocol that they are running. Each configuration is only provided based on the relationship between network elements with respect to a specific network protocol. At a basic level, the IP network relationship is maintained based on the relationship between AS, IP subnets and IP interfaces of routers. The IP network entities at the network layer are defined to be distributed entities. Similarly for OSPF if a parameter value is required to be same across all routers in a particular area, then IPNC will present this parameter at area level. Modifying it at area level will make sure that all routers in that OSPF area are updated. This prevents inconsistencies caused by today’s device centric configuration management.

2.3.1. Network Views

As mentioned earlier, the IPNC currently present three views of the network: IP network view, OSPF view and Device View.

2.3.1.1. IP Network View

The Figure 2.describes a hierarchical of view of the IP network. At the highest level of the hierarchy is the global IP network which is collection of AS. Each AS contains a set of subnets and connected by a set of routers. Each subnet contains a set of network interfaces of the associated routers that are in that subnet. Although, the routers are not strictly contained within the subnets, we show them just to associate the network interface to a router. Since a router may have interfaces in the multiple subnets, the same router may be shown multiple times.



```
IP Network
  AS 283
    135.180.172.88
      nc6.dnrc.bell-labs.com
    1.0.0.0
    135.180.172.56
      nc5.dnrc.bell-labs.com
        135.180.172.57
      nc7.dnrc.bell-labs.com
        135.180.172.58
    135.180.172.24
    135.180.140.24
    135.180.172.128
      nc1.dnrc.bell-labs.com
    192.168.252.0
    135.180.172.48
    10.2.1.0
      nc3.dnrc.bell-labs.com
        10.2.1.1
    135.180.172.8
    10.1.1.0
    135.180.172.40
    135.180.172.16
    135.180.172.64
    135.180.172.32
    192.168.253.0
```

IP Network – Interface

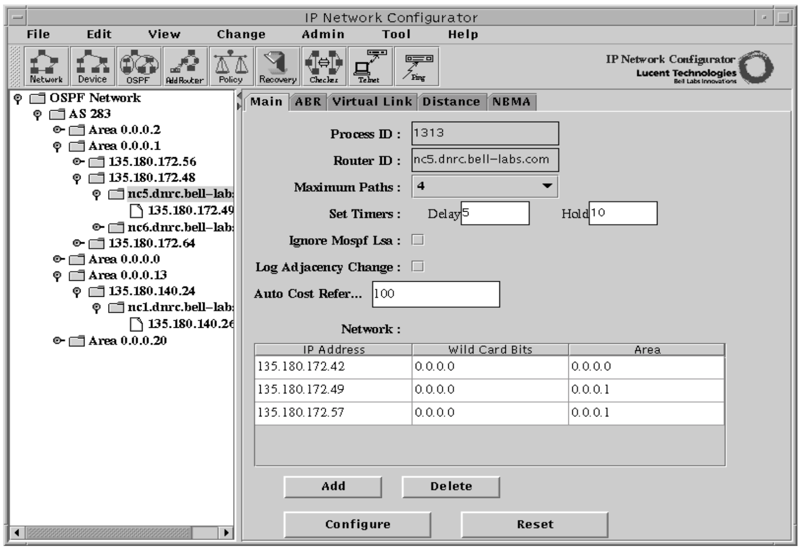
Interface Id : 135.180.172.57
Subnet Mask : 255.255.255.248
Link Config :

Parameters	Value
Type	
Speed	0
Physical Address	0010.7b87.6e22
Admin Status	1
Oper Status	1

Figure 2. The IP Network View

2.3.1.2. OSPF Network View

The Figure 3 describes the OSPF network view. It shows only the network that is currently in OSPF domain. As a result it only includes routers that are OSPF enabled. In addition, this view also imposes the OSPF hierarchy, which is key to enforce valid and consistent configuration updates in OSPF domain. User can view particular node of the tree and update corresponding entity. For example user can go to a particular area and modify authentication type used in the area. This will ensure that all routers in the area are updated with new configuration.



```
OSPF Network
  AS 283
    Area 0.0.0.2
    Area 0.0.0.1
      135.180.172.56
      135.180.172.48
        nc5.dnrc.bell-lab:
          135.180.172.49
        nc6.dnrc.bell-lab:
      135.180.172.64
    Area 0.0.0.0
    Area 0.0.0.13
      135.180.140.24
        nc1.dnrc.bell-lab:
          135.180.140.26
    Area 0.0.0.20
```

Process ID : 1313
Router ID : nc5.dnrc.bell-labs.com
Maximum Paths : 4
Set Timers : Delay 5 Hold 10
Ignore Mospf Lsa : ☐
Log Adjacency Change : ☐
Auto Cost Refer... 100

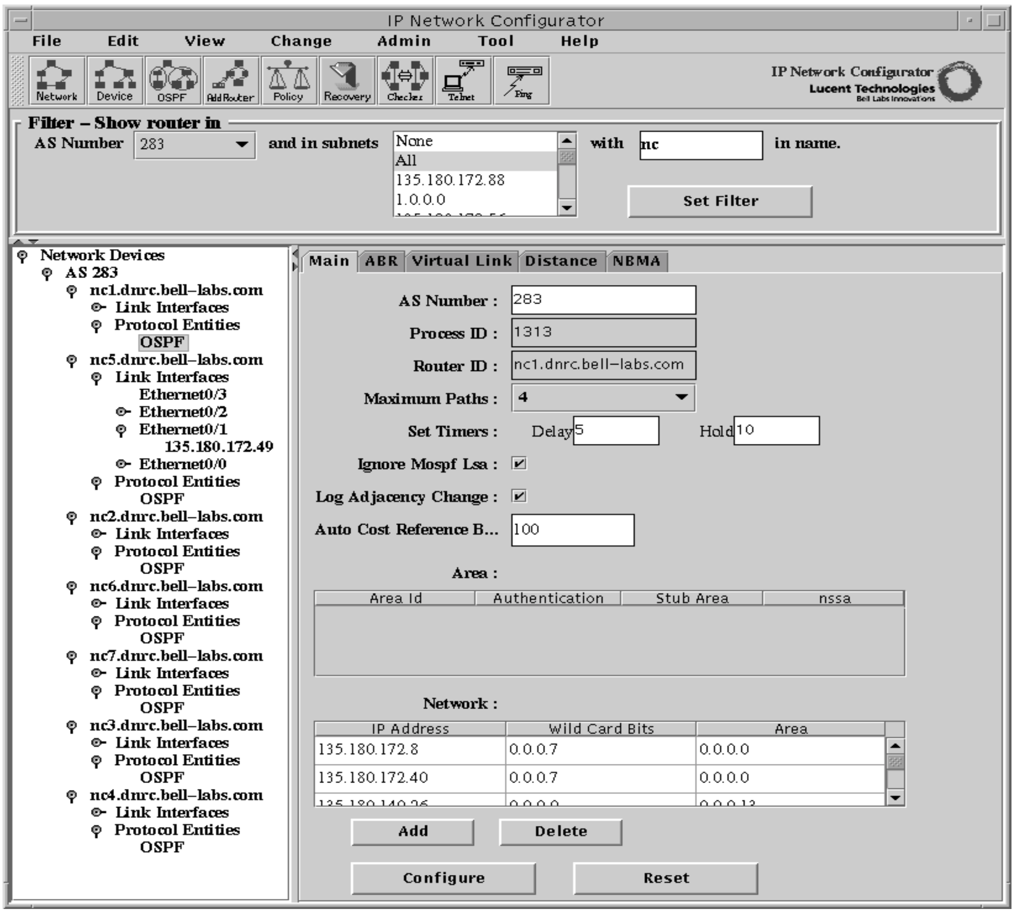
Network :

IP Address	Wild Card Bits	Area
135.180.172.42	0.0.0.0	0.0.0.0
135.180.172.49	0.0.0.0	0.0.0.1
135.180.172.57	0.0.0.0	0.0.0.1

Figure 3. The OSPF Network View

2.3.1.3. Device View

The Figure 4 describes the device view. The device view is a typical way the network elements are managed in commercial network management platform. In the device view no network level aggregation is shown. However, we do provide mechanism by which user can filter the content of the view. All the routers are aggregated at the AS level. For each router, the link interfaces and protocols are aggregated in two separate groups. For each link interface, all the IP network interfaces are shown. For each protocol that the router supports an entry is displayed under the protocols group.



Filter – Show router in
AS Number 283 and in subnets None / All / 135.180.172.88 / 1.0.0.0 with nc in name.

```
Network Devices
  AS 283
    nc1.dnrc.bell–labs.com
      Link Interfaces
      Protocol Entities
        OSPF
    nc5.dnrc.bell–labs.com
      Link Interfaces
        Ethernet0/3
        Ethernet0/2
        Ethernet0/1
          135.180.172.49
        Ethernet0/0
      Protocol Entities
        OSPF
    nc2.dnrc.bell–labs.com
      Link Interfaces
      Protocol Entities
        OSPF
    nc6.dnrc.bell–labs.com
      Link Interfaces
      Protocol Entities
        OSPF
    nc7.dnrc.bell–labs.com
      Link Interfaces
      Protocol Entities
        OSPF
    nc3.dnrc.bell–labs.com
      Link Interfaces
      Protocol Entities
        OSPF
    nc4.dnrc.bell–labs.com
      Link Interfaces
      Protocol Entities
        OSPF
```

AS Number : 283
Process ID : 1313
Router ID : nc1.dnrc.bell–labs.com
Maximum Paths : 4
Set Timers : Delay 5 Hold 10
Ignore Mospf Lsa : ✓
Log Adjacency Change : ✓
Auto Cost Reference B... 100

Area :

Area Id	Authentication	Stub Area	nssa

Network :

IP Address	Wild Card Bits	Area
135.180.172.8	0.0.0.7	0.0.0.0
135.180.172.40	0.0.0.7	0.0.0.0

Figure 4. The Device View

2.3.1.4. View Browsers

As a convenience for the programmer, we provide a set of view browsers for each of the view. The view browser filters out the routers and sub network that do not fit the specific view from the generic IP network view. The view browsers are registered with the RMI registry with a well-defined URL such that they can be found globally using the URL.

2.4. Configuration Service Layer

The configuration service layer defines one distributed entity for each of the IP protocol or service need to be configured, in addition to a set of common services required by the configurator. The configuration service layer provides the “business logic” for the configuration of the IP protocols and services. The entities for each of the configuration service are defined to be distributed entities. The configuration specific services are located using a service finder mechanism. The current IPNC tool defines following configuration services: OSPF configuration, versioning of router configuration. We also have partial support for RIP and BGP. The actual configuration related application is implemented at this layer and it provides an API for GUI based application or any other applications to invoke the service.

2.5. Configurator GUI

Finally, the configurator applications are defined as a set of web enabled workflow applications. These workflow applications are designed based on the configuration of the specific protocols. The front-end for these applications are a set of HTML pages containing applets. These HTML pages are downloaded from the WEB server, which is co-located with the IPNC server. The applets use RMI to communicate with objects in the IPNC server. The GUI provides capabilities like protocol specific views of the network to provision protocol specific tasks, front end for configuration consistency checking of the network, front end for topology discovery, front end for versioning/backup/restoration of the router configuration files etc.

3. IPNC Implementation Architecture

In Figure 5 the implementation architecture of the IPNC is described. Almost entire implementation of the IPNC architecture is based on the Java technology. Entities in various layers communicate with each other using Java RMI. The Device, Network and Configuration service layer

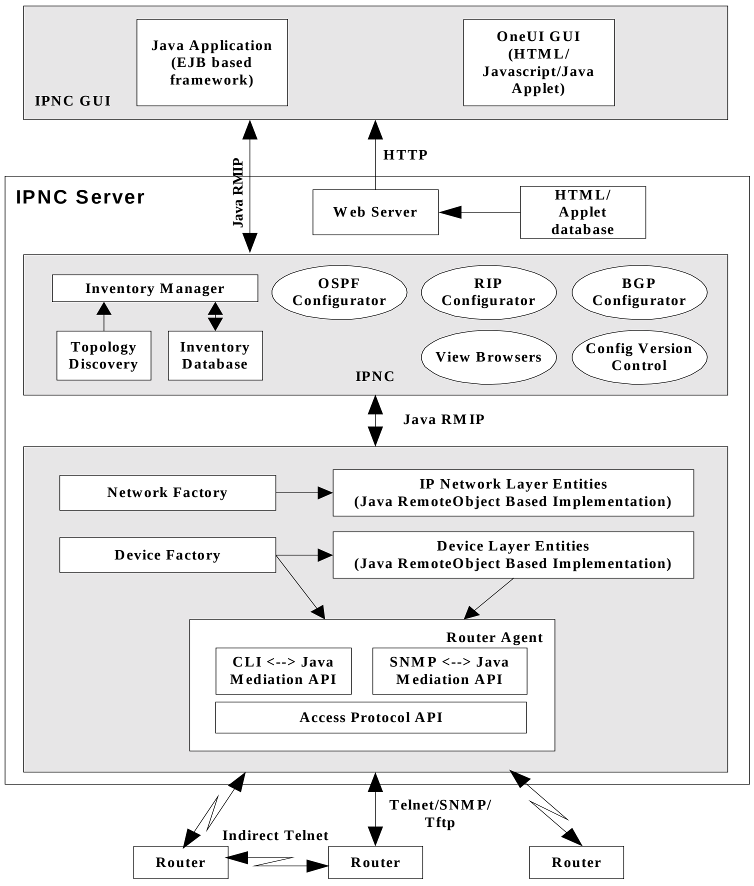


Figure 5. Implementation Architecture of the IPNC

entities are implemented in a single server framework, i.e., all the entities run in a single server or process though we use multi threading inside.

The *RouterAgent* object implements the entities in the DeviceAccess and the Device Mediation layers. The *InventoryManager* object along with the Topology Discovery Inventory Database implements the Inventory service of the Device Model layer. The DeviceFactory and NetworkFactory map to the *DeviceFactory* and *NetworkFactory* objects respectively. The entities of the Device and Network Model are implemented as remote Java objects.

When server starts-up, the *NetworkConfigurator* object is initialized which in turn initializes the *InventoryManager*, *DeviceFactory*, *NetworkFactory* objects. The *NetworkConfigurator* object then uses the *NetworkFactory* to initializes the server information based on the names of the router in the inventory. The Network Factory uses the Inventory Manager to read its inventory database. For each router in the inventory database, the *NetworkFactory* then creates and initializes an instance of a router using the *DeviceFactory*. Before each instance of the router is created, the network creates an instance of the network entities for each the network (As, subnet, area etc.) to which the router belongs. As the routers are initialized, a graph of the IP and OSPF network is incrementally built based on the information available in the routers.
The network view browser objects provide complete navigability to all of the objects using any of the following supported views: IP, OSPF, Device. The application logic for the OSPF, RIP and BGP configurator objects is implemented in the network and device entity directly and as such there are no specific configurator objects.

4. Information Model of the Router Agent Components

The architecture of the Router agent as shown in Figure 6 consists of four layers. Vendor independent Java API for accessing the router, an access method selection layer, protocol adapters and message mediators. The access method selection layer provides a mechanism to select a specific mediator (CLI, SNMP etc.) and protocol adapter (telnet, indirect telnet, tftp, SNMP, RMI, CORBA etc.). The message mediator maps java API calls to device specific message format. Specific protocol adapters are dependent on the protocol interface provided by the actual routers. Currently most routers provide Telnet and SNMP. Few vendors provide Java VM and ORB. The IPNC can be easily extended to support these vendors.

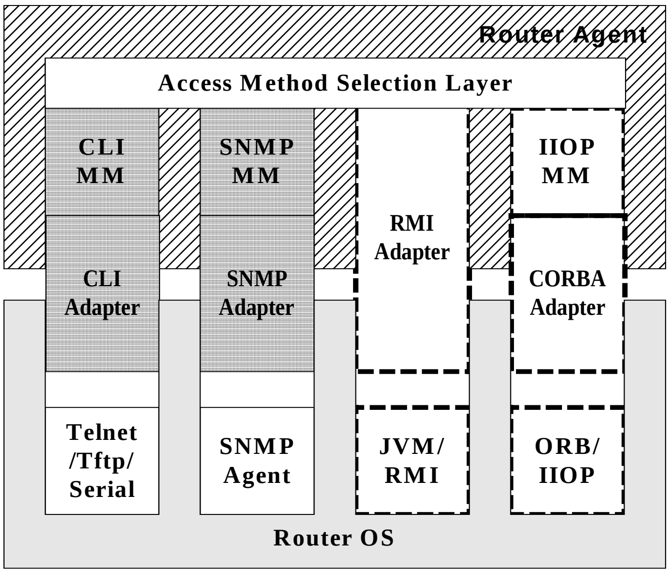


Figure 6. The Architecture of the Router Agent

The Figure 7 shows the various components of the router agent and their relationship. The *RouterAgentFactory* creates a *RouterAgent* object for specific router vendor. The *RouterAgentFactory* uses the vendor id of the router in the *RouterInvEntry* to select a specific type of router agent. The *RouterAgent* is comprised of following entities: *RouterInvEntry, RouterInfoAgent*, one or more mediator, one or more protocol adapters for each type of mediator, and *RouterConfigAgent* objects for configuration of the router.

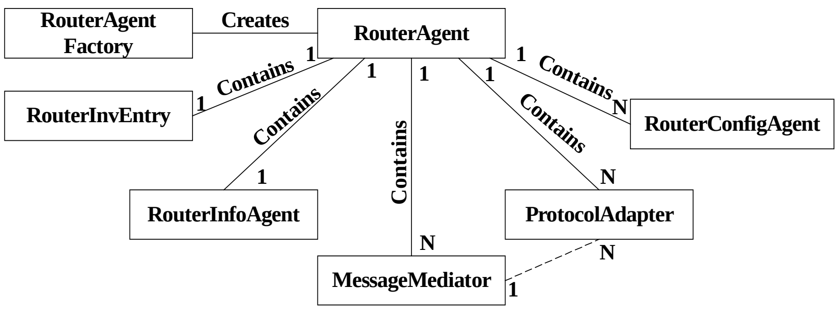


Figure 7. Information Model of Router Agent Components

The *RouterAgentFactory* creates an instance of the *RouterAgent* object and pass the *RouterInvEntry* reference to the *RouterAgent*. The *RouterAgent* then instantiates its other components based on the information in the *RouterInvEntry* object. It creates one instance of specific type (telnet, indirect telnet, SNMP, etc.) of protocol adapter. It also creates instance of each type of message mediator (CLI, SNMP etc.) and then associates message mediator to the corresponding protocol adapter. The *RouterInfoAgent* object represents the cached information of the remote Router. The references to the *RouterConfigAgent* are obtained by invoking *get<Protocol>RouterAgent()* method. The *RouterConfigAgent* provides both buffered and unbuffered write access to the remote Router. Figure 8 describes a specific instance of RouterAgent and its components.

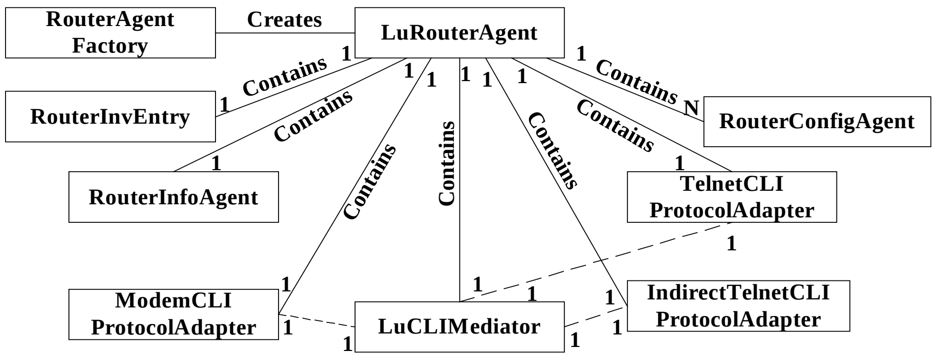


Figure 8. A Specific Instance of Router Agent and its Components

4.1. Inheritance Hierarchy of the Router Agent Components

Figure 9 describes the protocol/vendor specific extension of *MessageMediator* objects. The *MessageMediator* objects convert router information in Java format to the protocol specific message

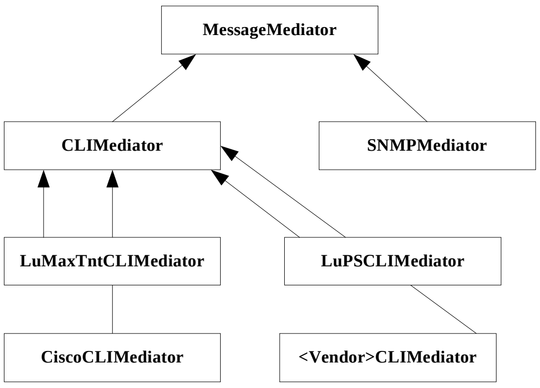
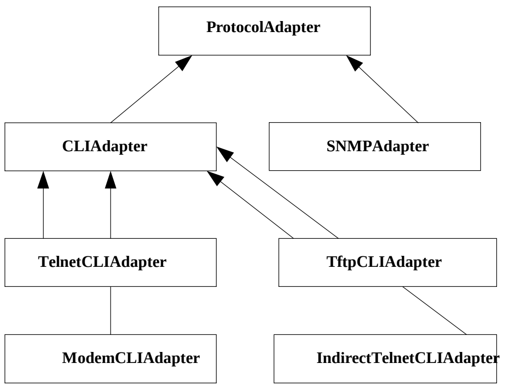


Figure 9. The Inheritance Hierarchy of the Message Mediators

format and vice versa. The *MessageMediator* is first extended for each type of message format, e.g. CLI and SNMP. Then it is extended for each vendor's product. A vendor may have one or more *CLIMediator* extensions depending on how many different types of CLI does a vendor support. This extensibility is very useful in today's dynamic environment where acquisition of a company brings a product with different management interface or format. SNMPMediators can also be extended to support version specific behaviors. It is also possible to use different types of mediator for same device e.g. SNMPMediator for gets and CLIMediator for sets. The CLIMediators essentially converts Java name value pair list to vendor dependent CLI commands and vice versa e.g. `ospfIf.setOspfHelloInterval("10")` Java RMI call is translated to `CiscoCLIMediator.setRouterIf(Properties nvp)` where nvp is a name value pair list containing SNMP equivalent name `ospfIfHelloInterval` and value 10. This in turn executes "`ip ospf hello-interval 10`" CLI command on a router in appropriate configuration context. Similarly SNMPMediator converts `SNMPMediator. setRouterIf (Properties nvp)` to same command.



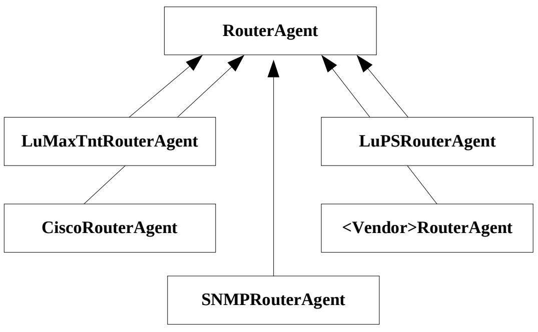


Figure 10. The Inheritance Hierarchy of the Protocol Adapters

Figure 11. The Inheritance Hierarchy of the Vendor Dependent RouterAgent

Figure 10 illustrates the inheritance hierarchy of the access protocol specific ProtocolAdapters. The ProtocolAdapters transparently manages the connection setup, termination and messaging between the RouterAgent and the remote Router. The flexibility to separate management data and access mechanism allows IPNC to try different access method if one fails e.g. if direct telnet to router fails then IPNC tries to reach router through indirect telnet or modem. This fail over scheme is transperant to user.

Figure 11 illustrates the inheritance hierarchy of the vendor specific *RouterAgent*. The implementation of the vendor specific router agent transparently assigns appropriate *MessageMediator*. The RouterAgent creates an instance of specific protocol adapter based on the access information in the *RouterInvEntry*.

5. Conclusion

We believe that with the rapid emergence of new services and new vendors it is very important to build a network and service-provisioning infrastructure that can allow incremental support for these new requirements. The infrastructure has to be scalable, extendible and based on standard technologies. The IPNC provides this software infrastructure and it can meet the challenges of tomorrow. One of the important issue that IPNC addresses is access and mediation for different devices in a transparent manner. It is also very easy to extend IPNC to support different vendors and new services. Since IPNC is based on Java it can support several platforms without any change in implementation. We believe that availability of standard based service models in future can make this type of infrastructure even more versatile.

6. Acknowledgement

We would like to thank IPNC team at Bell Labs and Lucent business unit for their support in building this tool.

7. References

[1] P. Krishnan, Tejas Naik, Ganesan Ramu and Roshan Sequeira, "Sequencing of Configuration Operations for IP Networks", proceedings of the 14th Systems Administration Conference, pp 265-274, December 2000.

[2] Lucent Technologies, “IP Network Configurator”, http://www.lucent.com/OS/ipnc.html

[3] M. MacFaden, and J. Saperia, “Configuring Networks and Devices with SNMP”, draft- ietf-snmpconf-bcp-01.txt, Inc, May 3, 2000.

[4] H. Ku, J. Forslow, and J.Park, “Web-based Configuration Management Architecture for Router Networks”, J.W. Hong and R.Weihmayer (ed.), proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2000), pp 173-186, April 2000.

[5] Astracon Inc, http://www.astracon.com/

[6] Cisco Systems, NETSYS Connectivity Tools Reference Manual, http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/netsys/netsysrg/index.htm.

[7] Cisco Systems, “Cisco IP Manager”, http://www.cisco.com/warp/public/cc/pd/nemnsw/ipmn/prodlit/ipman_ds.htm

[8] Orchestream Inc, http://www.orchestream.com/

[9] Syndesis Limited, http://www.syndesis.com/

[10] Vertel Corporation, http://www.vertel.com/