Establishing an Architectural Model

By John V. Nguyen - Sun Professional Services
Sun BluePrints™ OnLine - February 2002

http://www.sun.com/blueprints

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 USA
650 960-1300

Part No.: 816-4597-10
Revision 1.0, 02/06/02
Edition: February 2002

Establishing an Architectural Model

Editor's Note – This article is the complete third chapter of the Sun BluePrints™ book, "Designing ISP Architectures", by John V. Nguyen (ISBN 0-13-045496-6), which is available through www.sun.com/books, amazon.com, fatbrain.com and Barnes & Noble bookstores.

This chapter introduces an architectural model as a framework for designing an ISP architecture. The model is based upon our experience and Sun best practices for designing ISP architectures. For architects who want it, the first two sections of this chapter provide background and definitions of concepts and components necessary for understanding an architectural model.

This chapter contains the following topics:

Also, building upon the information in Chapter 2, this chapter shows how to apply the architectural model and principles to design requirements.

Understanding the Model

An ISP architectural model provides a design framework for ISP architectures, which are often complex and comprised of multiple components requiring careful consideration and design. When designing an architecture, it is helpful to use or create a model, then apply all the requirements, assumptions, and design trade offs.

The model presented here is from our point of view, based upon experience in design and resulting best practices. Although there are many other architectural models, principles, and ways of approaching a design, for purposes of demonstration we focus on selected key components and principles. We advise you to determine which attributes and principals are most appropriate for your design, from a larger pool of architectural design standards.

FIGURE 1 shows a sample architectural model. In the center of the model are key components. Surrounding these key components in smaller circles are architectural principles.

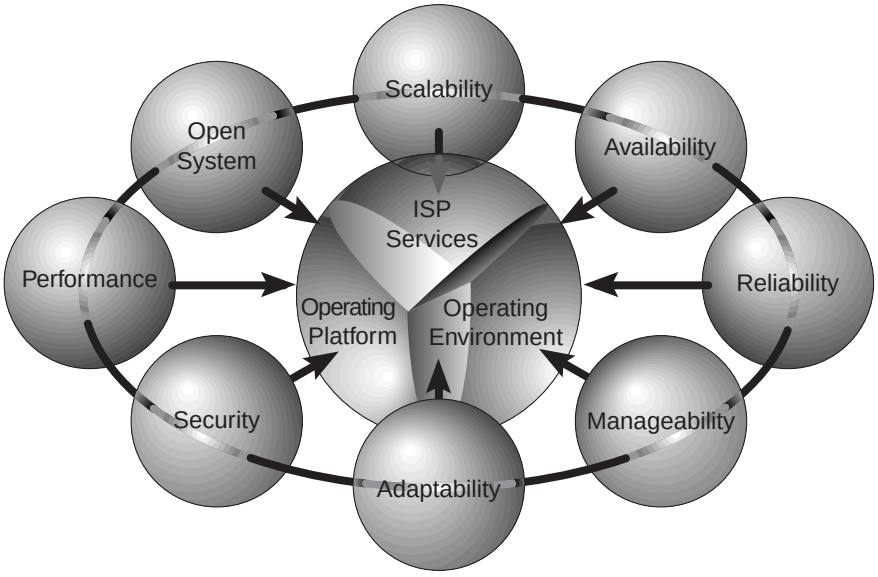

FIGURE 1 ISP Architectural Model

As shown in the figure, key components serve as the core for the architectural design. In the outer layer, architectural principles provide structure and considerations for making design decisions, then adhering to a design.

Identifying Key Components

After analyzing business and functional requirements (see Chapter 2), the initial step in modeling an ISP architecture is to identify key components.

In general, key components are uniform among most designs; at the minimum, there should be ISP services running within an operating environment on an operating platform. As shown in FIGURE 1, the core of this model consists of ISP services, operating environment, and operating platform.

Tip – The differences in a design are typically in the selection of services, an operating environment, and the operating platform, all of which are based on business requirements and preferences.

ISP Services

ISP services are usually categorized into four types: basic services, value-added services, infrastructure services, and operation and management services. The following paragraphs describe each of these.

Basic Services

Basic services are common services offered by ISPs to residential and business subscribers. As shown in FIGURE 2, basic services are email, web hosting, and Internet news. Although not listed, Internet access and FTP (file transfer protocol) are considered basic services; they are required for connectivity and content uploads, respectively.

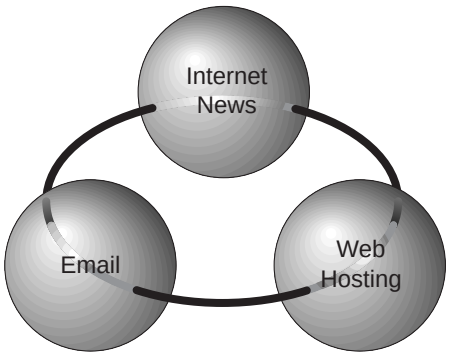

FIGURE 2 Basic Services

Internet News	Post news on the Internet
Web Hosting	Host personal web pages
Email	Send and receive email

Value-Added Services

Value-added services are special services offered to provide additional value to existing subscribers, to attract new subscribers, and to differentiate services from those offered by competitors. FIGURE 3 shows a sample of value-added services an ISP might offer.

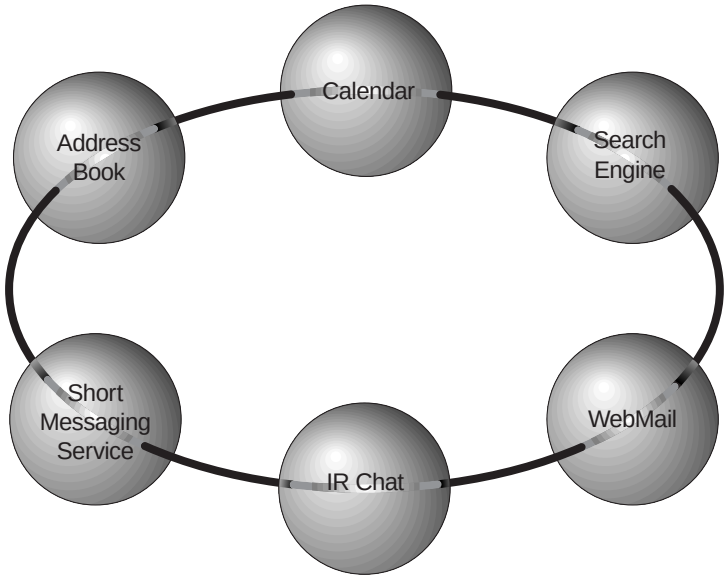

FIGURE 3 Value-Added Services

Calendar	Schedule appointments
Search Engine	Online search capabilities
WebMail	Email via web browser
IR Chat	Internet relay chat (IRC)
Short Messaging Service	Send text messages via short messaging service (SMS)
Address Book	Personal address book

What constitutes value-added services varies among ISPs and changes quickly as competitors follow leaders. Samples of value-added services are calendar, search engine, WebMail, IRC, SMS, and address book.

To add value, these services enhance a user's experience and provide tools that users want conveniently at their fingertips. Large ISPs today are aiming to be one-stop portals for everything from web surfing to online shopping.

As new services become more common, many ISPs subsequently convert value-added services to basic services.

Infrastructure Services

Infrastructure services are services that are absolutely critical to support other ISP services running within an infrastructure. These services run in the background and are transparent to users. Infrastructure services are the workhorses of infrastructure functions. FIGURE 4 shows the minimum required infrastructure services.

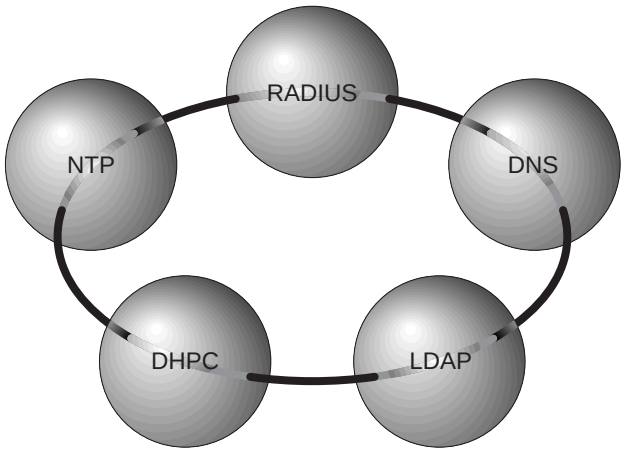

FIGURE 4 Infrastructure Services

DNS	Domain name system is for name resolution.
LDAP	Lightweight directory access protocol (LDAP) is for authentication and authorization.
RADIUS	Remote access dial-in user service (RADIUS) is for remote access authentication.
NTP	Network time protocol (NTP) is for time synchronization.
DHCP	Dynamic host configuration protocol (DHCP) is for dynamic host configurations for client systems.

Operation and Management Services

Operation and management services are services that allow system administrators to maintain an environment and provide business continuity through uptime. These services are critical to the operation and management of an ISP. Routine tasks such as performing nightly backups, changing tapes, restarting services, installing software patches and upgrades, and monitoring ensure that the environment is working well.

FIGURE 5 shows operation and management services. Although these services are technically a form of infrastructure services and play a support role within an infrastructure, one or more of these services might not be an absolute requirement, depending upon an ISP's business requirements.

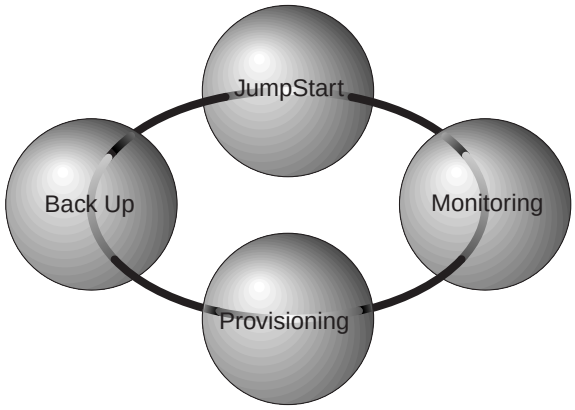

FIGURE 5 Operation and Management Services

JumpStart	Automates system installation and management tasks.
Monitoring	Monitors system utilization, intrusions, service availability, etc.
Provisioning	The two categories of provisioning are user and services. User provisioning consists of new user registration, care, and billing. Service provisioning consists of installing new software, patch updates, and software upgrades.
Back Up	Nightly backup for data protection and disaster recovery.

Operating Environment

An operating environment (OE) consists of an operating system (OS) and bundled tools and applications to provide a total solution with seamless integration. Most vendors offer a wide selection of packages for their OS, with different tools and applications.

Note – Most Internet tools are developed in UNIX® before they are ported to other platforms, which may be a consideration when choosing an OE.

Most vendors include applications with an OE. These applications can be commercial, open source, or a combination of both. Commercial applications are usually high-end applications for enterprise environments, and licensing for these applications varies among vendors. Open source applications are usually lower-end applications with limited functionality and features, and licensing agreements are commonly provided under general public license (GPL).

Operating Platform

An operating platform is the underlying hardware platform that supports the operating environment. This hardware includes network equipment, enterprise servers, storage, etc.

Applying Architectural Principles

Supporting key components of the sample ISP architectural model are architectural principles, as shown earlier and again in FIGURE 6. Architectural principles are major design considerations that help you qualify advantages and disadvantages of each design option, so that you arrive at a solution that *best fits* business requirements, functional requirements, and available technology.

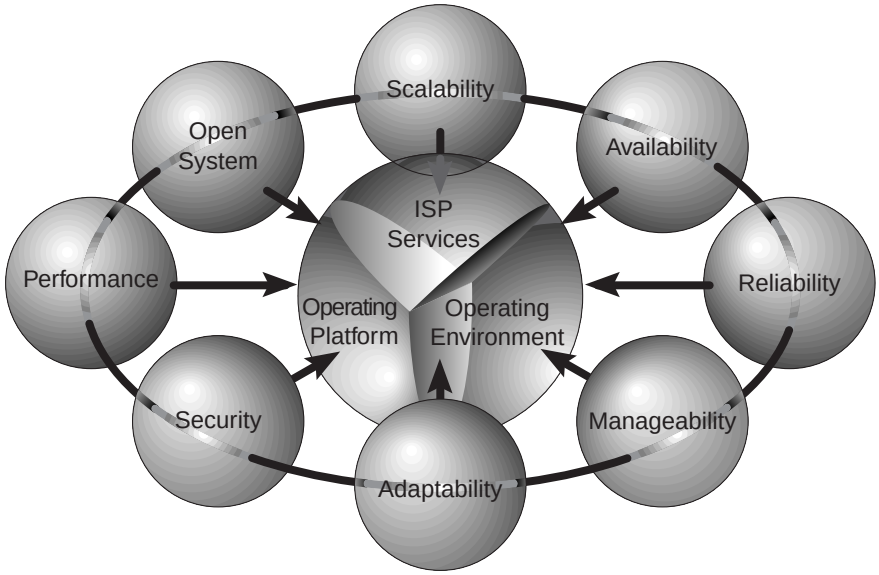

FIGURE 6 ISP Architectural Model

We categorize architectural principles into eight areas: scalability, availability, reliability, manageability, adaptability, security, performance, and open system. Although there are other design principles you might need to use or consider, we focus on these as most critical.

Consider each of these principles (and any others that apply) when evaluating design issues and trade-offs for key components. For example, apply scalability to different layers within an architecture. You could address it at the network, system, and application layers. Failing to address scalability at each layer could result in nonoptimal scalability for an architecture.

Ultimately, some architectural principles may not apply to your design. However, it's important initially to consider them as part of the design process, especially for large-scale environments with higher levels of complexity. For example, if cost is a significant design constraint, then adding expensive layers of redundancy to enhance availability is most likely not applicable.

Scalability

Scalability is the ability to add additional resources, for example, routers, switches, servers, memory, disks, and CPUs to an architecture without redesigning it. A good design takes into account the need for scalability so that, within reason, as a business grows and user demand increases, new computing resources can be added on demand. Some customers have a clear idea of their plans for growth and indicate such at the beginning, while others may need you to suggest and build in scalability, based upon your interpretation of their current and future business requirements.

When you address scalability, we recommend using the following scaling models, depending upon which one is applicable to your design. These are simplified models that address scaling for both hardware and software at the same time during the architecture design process.

TABLE 1 Scaling Model for Servers

Scaling Model	Vertical	Horizontal
System Type:	Single Large System	Multiple Small Systems
Software Type:	Multithreaded applications	Single-threaded applications
To Scale:	Add CPU, memory, disk, and I/O	Add additional systems

Both models apply to key components. Each major component within an infrastructure, for example, network, system, application, storage, etc., has its own scaling model.

Vertical Scalability

Multithreaded applications are more complex in their scaling model. Typically, the first line of scaling within a single system for a multithreaded application within a single system is to achieve the maximum vertical scalability by adding more resources such as CPU, memory, and I/O. Vertical scaling is appropriate for applications that scale well within a single large server, such as database servers.

Tip – Scale multithreaded applications vertically first. When maximum vertical scaling is achieved, scale the same applications using horizontal scaling techniques, for example, running the applications on multiple boxes behind a load balancer.

Horizontal Scalability

For single-threaded applications, the model for scaling is horizontal. In this model, a vertical scaling limitation of the server is replaced with a much more scalable load distribution paradigm. This technique is deployed at a system level by adding more servers to increase scalability.

Tip – Unlike multithreaded applications, single-threaded applications do not achieve optimal benefits from vertical scaling. For example, adding more memory benefits single-threaded applications; however, adding another CPU does not. Scaling horizontally can be done by running multiple instances on multiple boxes behind a load balancer.

In contrast to availability, which is designed for failover, the purpose of multiple system redundancy in scalability is to provide a model for adding resources to increase capacity.

Availability

Availability has many definitions within Internet architectures. In this book, it means that resources and access to those resources are available upon request. Availability design is predicated on the removal of any single point-of-failure within an architecture to ensure a desired level of uptime. This uptime is usually expressed in percentages and often referred as the "number of 9s." For example, most mission critical systems have a desired uptime of "five 9s," meaning that the system is available 99.999 percent of the time.

TABLE 2 Availability Levels

Uptime Percentage	Nines	Allowable Downtime Per Month
99.9999	6	0.043 minute
99.999	5	0.43 minute
99.99	4	4.30 minutes
99.9	3	43 minutes
99	2	7.2 hours

We determined allowable downtime by using the following formula:[1]

$$Availability = \frac{MTBF}{MTBF + MTTR}$$

where MTBF is mean time between failure and MTTR is mean time to repair.

For marketing reasons, many ISPs calculate the level of availability over a 12-month period instead of monthly. (This practice yields an overall higher average level of availability than calculating it monthly, because monthly calculations fluctuate from month to month.)

We calculate the availability monthly because system administrators typically perform maintenance monthly; therefore, monthly calculations are more beneficial for determining allowable downtime to perform maintenance and upgrades. This practice is fairly universal for system administrators of ISPs. Other reasons for calculating it on a monthly basis:

- Revenue, usage, stats, spending, etc. are done monthly.
- Waiting for one year to find out the level of availability is unrealistic.

1. Priscilla Oppenheimer, *Top-Down Network Design*, Cisco Press®, 1999.

A primary attribute of availability design is redundant hardware/software within the architecture, such as network, server, application, and storage.

Tip – Design in such a way that if a component fails, it does not cause the entire architecture to fail. To achieve this design objective, design using a modular approach, allowing components to be replaced at any time without affecting the availability of the system.

The four layers, covered in the following paragraphs, are as follows:

- Network layer
- System layer
- Application layer
- Data layer

Network Layer

At the network layer, availability can be achieved with redundant physical links to the Internet. This redundancy ensures that if there is a link failure, for example, due to hardware failure, access is still available via a surviving link. In addition, redundant network components such as routers, switches, load balancers, and firewalls are necessary to ensure access availability in the event of hardware failure. To enhance reliability at the network layer, remove all single points-of-failure from the network.

Note – For the Solaris™ Operating Environment (Solaris OE), IP multi-pathing (IPMP) can be used to achieve redundant network connections from the same server to multiple switches.

System Layer

At the system layer, availability is achieved with redundant servers in stand-alone or cluster configurations.

For front-end servers such as those deployed in web farms, you can use load balancers to ensure availability in the event that one or more servers fail to respond to service requests.

In a cluster environment, two or more servers are configured to provide high availability. The number of nodes configured in a cluster is dependent upon the software and hardware. If one server fails, one of the surviving servers takes over and responds to service requests.

A fundamental difference between stand-alone servers and clustered servers is the ability to maintain session states. If a stand-alone server fails while a session is active, the connection has to be reestablished from the client. However, if a clustered server fails, the session state and connection is maintained by a standby server.

Note – The cost of redundant servers and software licensing is extremely expensive for small- to mid-size ISPs. However, without it, ISPs may lose subscribers and revenue to competing ISPs because of subscriber dissatisfaction from service interruptions. Subscriber expectations for availability and reliability are usually high, and many competitors already offer high availability and reliability.

Application Layer

At the application layer, availability can be achieved with clustering and high availability software. You can configure applications with clusters or high availability to enhance availability in the event of service failure. Service failure and restart can be automatically invoked through service failure detection and monitoring. Also, you can enhance availability at the application layer by using a load balancer with multiple servers.

Data Layer

At the data layer, availability can be achieved with redundant storage arrays coupled with logical volumes. Redundant storage arrays allow data to be accessible in the event of a controller or storage array failure. Logical volumes and RAID (redundant array of independent disks) ensure data is accessible in the event of disk failure.

At the data layer, RAID 0+1 (stripping and mirroring) or RAID 5 (stripping with parity) achieves availability and reliability in case of disk failure. RAID 0+1 is a more expensive solution because twice the hardware (storage arrays and disks) is needed. However, the advantage is that no performance degradation occurs due to a disk failure. RAID 5 can have performance degradation if a disk fails, because data has to be rebuilt from parity.

Reliability

Reliability is best defined from the perspective of end users. Users want network services and servers to be available when they access them. Reliability for them is consistency of service uptime and availability. To users, a system is reliable when they do not frequently encounter busy signals on their modems, network connection error messages, etc.

From an architect's perspective, reliability is uptime and service response time for users, so that a system is available when users access services.

For businesses today, especially service providers, reliability of service has implications beyond customer satisfaction. Because service providers establish and maintain their reputations based on availability and reliability of their services, many of them require carrier-class grade high availability and reliability.

Tip – Reliability depends upon and is affected by the design for availability; therefore, your design for an ISP architecture should balance a customer's requirements for both availability and reliability, within any constraints imposed by customer or technology.

Dependent upon availability design, reliability is increased through an infrastructure based on redundant servers. Functionally componentized architecture results in more intrinsic redundancy and fewer inherent single points-of-failure. Furthermore, any damage to an individual service is unlikely to impact other services.

The constructs of redundancy are useful in achieving many aspects of reliability, scalability, and availability.

Manageability

Manageability addresses how an infrastructure can be managed during its life cycle. The key to manageability is to keep an architecture design simple, yet effective. Meet all functional and business requirements without adding complexity. If a design is too complex and difficult to manage, there is more likelihood for operation and management failure, and troubleshooting becomes more difficult and time consuming. Also consider management tools, management plans, and methods of monitoring services. Ensure that devices and components that need to be monitored are managed. If a system goes down and there is nothing monitoring the device or component causing the outage, customer satisfaction and subscriber satisfaction are at risk, in addition to associated costs and potential loss of revenue.

Adaptability

For any architecture, change during a life cycle is inevitable. An architecture must be adaptable enough to accommodate growth and changes in technology, business, and user needs. Within the customer's financial constraints and growth plans, design an architecture that allows for adaptability.

Modular architectures inherently support flexibility in two ways: individual components are themselves easily augmented, and, because components are independent, new components can be added without disturbing or revamping other components within an architecture.

Security

From a larger perspective, security is a combination of processes, products, and people. Security is achieved by establishing effective policies and implementing procedures that enforce policies. Security policies are useless without control over who has access to and can affect security on servers and services. Securing access requires establishing an appropriate authentication regime.

From an architecture perspective, security is access to network, system, and data resources.

- At the network layer, security can be achieved with an access control list (ACL) on routers, packet filters, firewalls, and network-based intrusion detection systems (IDS).
- At the system layer, security can be achieved with system hardening, access permission, host-based IDSs, scanners, and file checkers.
- At the data layer, security can be achieved with authentication and authorization.

Functional decomposition (separating functional components) contributes to security by making it easy to build security around different components. In addition, if one component is compromised, the security breach may be more easily contained.

Adapting to evolving threats is a never-ending cycle of processes. The strategy of responding to security threats has to evolve as potential intruders gain knowledge and discover new attack techniques.

We recommend designing security strategies with great flexibility in approaches to provide the best security against present and future threats.

Performance

Although performance has multiple definitions, in this book we relate it to the “expected” response time after a user requests a service. Depending upon an ISP’s requirements, response time may be critical or noncritical, and these distinctions may be further refined by service type.

Individual services use system resources, for example, memory, CPU, and I/O, in different ways. A modular architecture provides the ability to independently monitor and tune each service.

The causes of slow response times are many. For example, some common causes are network latency, server degradation, and application responsiveness. Degradation at any of these layers can result in poor overall performance.

A system is easier to tune when it is running only a few applications. When many applications are running on a system, they must share resources, and tuning becomes complicated and challenging.

Tip – Two products available from Sun are useful in managing resources: Solaris Resource Manager and Solaris Bandwidth Manager. The Solaris Resource Manager manages resources for users, groups, and enterprise applications. The Solaris Bandwidth Manager controls bandwidth allocated to applications, users, and organizations.

Open System

Ideally, design using an open system approach so that an architecture is not dependent upon a single hardware or software vendor. An architecture is less flexible when built upon proprietary specifications. Building upon a set of open system standards that are accepted by a recognized consortium provides greater flexibility for business changes and growth, such as adding users and services and integrating new technology.

Applying the Model to FijiNet

In this section, we apply the architectural model to FijiNet. We combine the requirements, assumptions, and evaluation we formulated in Chapter 2 with the model and principles presented in this chapter.

Identify Key Components for FijiNet

After evaluating the business and functional requirements, we identify key components for FijiNet and find that they are in line with general design for ISP architectures. The core components consist of ISP services, an operating environment, and an operating platform.

ISP Services

FijiNet ISP services are basic, infrastructure, and operation and management services. While value-added services are not offered initially, they may be offered in the future.

Basic Services

FijiNet wants to offer basic services to residential subscribers. These services are email, web hosting, and Internet news. FTP is available for content uploads.

Value-Added Services

FijiNet is not offering value-added services initially; however, they might offer them in the near future.

Infrastructure Services

For FijiNet, all infrastructure services presented in the model apply (DNS, LDAP, RADIUS, DHCP, and NTP). These services represent the workhorse of FijiNet's infrastructure.

Operation and Management Services

Operation and management services for FijiNet are outsourced, including provisioning (billing, registration, and customer care).

Note – Operation and management services are beyond the scope of this book. Many resources are available, such as *OSS Essential: Support System Solutions for Service Providers*.

Operating Environment

The operating environment for FijiNet is a reliable operating system comprised of commercial and open source applications. To minimize cost, open source software is used as much as possible. TABLE 3 lists components for FijiNet's operating environment.

TABLE 3 Operating Environment for FijiNet

Product	Type	Description
Solaris 8 Operating Environment	Commercial	Operating system
Cisco PIX	Commercial	Firewall appliance
Solstice Backup™*	Commercial	Backup software (bundled with Solaris 8 OE); free usage for up to 200,000 entries
Amdocs Horizon (formerly Solect IAF Horizon)*	Commercial	Billing system for service providers
iPlanet™ Directory Server	Commercial	Directory software (bundled with Solaris 8 OE); no charge for single server licenses
Steel-Belted Radius	Commercial	RADIUS software for service providers
DNS	Open source	DNS software (free with Solaris 8 OE)
DHCP	Open source	DHCP software (free with Solaris 8 OE)
NTP	Open source	NTP software (free with Solaris 8 OE)
sendmail	Open source	Mail software (free with Solaris 8 OE)
WUftp	Open source	LDAP-compliant FTP software (free/bundled with Solaris 8 OE)

TABLE 3 Operating Environment for FijiNet *(Continued)*

Product	Type	Description
WUimap	Open source	POP/IMAP (post office protocol/Internet mail access protocol) Internet mail software (free/bundled with Solaris 8 OE)
Apache	Open source	Web software (free/bundled with Solaris 8 OE)
INN*	Open source	News software (free/bundled with Solaris 8 OE)
OpenSSH	Open source	Secure SHell software

*These products are applicable if an ISP manages billing and news services internally. If an ISP chooses to outsource these services, then these software products are not needed.

Operating Platform

The operating platform for FijiNet is comprised of high-performance enterprise equipment (network, server, storage, etc.). Hardware was chosen based on FijiNet's requirements and cost constraints. The hardware supports an initial 10,000 subscribers and is scalable to 100,000 subscribers. TABLE 4 lists components for FijiNet's operating platform.

TABLE 4 Operating Platform for FijiNet

Product	Vendor	Description
Enterprise server	Sun Microsystems	Netra™ t1
Enterprise server	Sun Microsystems	Ultra™ 280R
Enterprise storage	Sun Microsystems	Sun StorEdge™ D1000
Enterprise library	Sun Microsystems	Sun StorEdge L280
Router	Cisco Systems	Cisco 2651
Switch	Cisco Systems	Cisco 3512XL
Firewall	Cisco Systems	Cisco PIX 525
Access server	Cisco Systems	AS5400
Console server	Cisco Systems	AS2511

Note – The quantity for each component is provided with capacity planning in Chapter 6.

Apply Architectural Principles to FijiNet

We apply each of the principles to FijiNet's requirements, our interpretation and assumptions, and our evaluation. For detailed information supporting each of the principles applied, refer to Chapter 2.

Scalability

Due to cost constraints, we specify the smallest possible hardware that can handle the load. The architecture scales horizontally. Because of the smaller chassis size, the system has limited vertical scalability. For scaling from 10,000 to 100,000 subscribers, horizontal scaling is much more economical and flexible.

Availability

No redundancy is implemented, due to cost constraints. To provide a higher level of data availability at an affordable cost, we implement RAID 0+1. FijiNet's business plan and case do not warrant investment in redundancy for high availability at this time.

Reliability

The hardware we specify, enterprise server and storage, are very reliable. Although the hardware reliability is very high, we acknowledge that a single chassis component could fail, because there is no failover.

Manageability

We settle on a 2-tier architecture for FijiNet, to simplify the design. Due to cost, a single-box solution is the best fit. We acknowledge that if FijiNet wants to implement an N-tier architecture later, they need to implement a new architecture. A 2-tier architecture with a single-box solution cannot be retrofitted or scaled to be an N-tier architecture.

Adaptability

The architecture is based on open standards. We use no proprietary technology; therefore, the architecture should be adaptable and integrate with any open systems, standards-based technology. Also, the design is modular and should be adaptable to changes with no reconfiguration or rearchitecting.

Security

An ACL and packet filters provide a basic front-end filter at the router. We use a premises firewall for access control. At the host level, operating system hardening ensures proper file permission. For the Solaris OE, the Solaris Security Toolkit (JASS) is available from Sun Microsystems for OS hardening.

Performance

Based on benchmark results for various infrastructure services such as DNS and firewall, we are confident that FijiNet's server can be load tested with a simulated load. (Refer to Appendix F for benchmark data.) Note that without real user profile and usage pattern data, it's hard to predict actual load.

Open System

The architecture design for FijiNet uses open systems hardware and software based on recognized industry standards.

Author's Bio: John V. Nguyen

John V. Nguyen is a Senior Architect at Sun Microsystems working in Advanced Internet Practice (Sun Professional Services). He has over 10 years experience in software engineering, systems management, networking, security, and architecture design for large-scale service providers, portals, and Internet datacenters. John is currently working on wireless technologies such as Location-Based Services, Mobile Messaging, Wireless Portal, 3G Wireless Networks, and Wireless Intelligent Networking. John holds a BS in Electrical Engineering, BS in Computer Science, and Ph.D. in Computer Science.