ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

UNIVERSIDAD

Computer Networks Laboratory

Laboratory No. 6
Application Layer Protocols

Web Server Installation, DNS Configuration, and Virtual Hosting

Students:

Andersson David Sánchez Méndez

Cristian Santiago Pedraza Rodríguez

Instructor: Professor Fabián Eduardo Sierra Sánchez
Course: Computer Networks

Institution: Escuela Colombiana de Ingeniería Julio Garavito

October 22, 2025

Contents

1 Objectives **4**

2 Tools and Equipment **4**
2.1 Required Software . . . 4
2.2 Server Infrastructure . . . 4

3 Introduction **4**

4 Part 1: Web Server Installation **4**
4.1 1.1 Apache Installation . . . 4
4.2 1.2 Creating Web Page . . . 4
4.3 1.3 Automatic Startup . . . 5
4.4 2.1 Nginx Installation . . . 5
4.5 2.2 Creating Web Page . . . 5
4.6 2.3 Automatic Startup Configuration . . . 5
4.7 3.1 IIS Installation . . . 5
4.8 3.2 Creating Web Page . . . 6

5 Part 2: DNS Configuration **6**
5.1 4.1 BIND Installation and Configuration . . . 6
5.2 4.2 DNS Zone Files . . . 6
5.3 4.3 Service Activation . . . 6
5.4 4.4 Client Configuration . . . 7

6 Part 3: Virtual Host Configuration **7**
6.1 5.1 Directory Structure . . . 7
6.2 5.2 Virtual Host Configuration . . . 7
6.3 5.3 Enable Virtual Hosts . . . 7
6.4 5.4 DNS Configuration for Virtual Hosts . . . 7
6.5 5.5 Testing Virtual Hosts . . . 8

7 Testing and Verification **8**
7.1 Connectivity Tests . . . 8

8 Infrastructure Summary **8**

9 Troubleshooting and Solutions **8**
9.1 Apache Virtual Hosts Issue . . . 8
9.2 DNS Resolution Delay . . . 8
9.3 Nginx Compilation . . . 9

10 Key Learnings **9**

11 Application Layer Protocols **9**
11.1 HTTP Protocol . . . 9
11.2 DNS Protocol . . . 9

12 Network Architecture **9**

13 Security Considerations **9**
13.1 Access Control . . . 9
13.2 Best Practices Implemented . . . 9

14 Performance Optimization **10**
14.1 Web Server Tuning . . . 10
14.2 DNS Optimization . . . 10

15 Comparison: Apache vs Nginx vs IIS **10**

16 Future Enhancements **10**
16.1 Potential Improvements . . . 10

17 Conclusions **10**

18 References **11**

1 Objectives

- Install and configure web servers on multiple operating systems (Solaris, Slackware Linux, Windows Server)
- Configure DNS service for domain name resolution
- Implement virtual hosting to serve multiple websites from a single server
- Understand application layer protocols (HTTP, DNS)
- Configure automatic startup and remote access for web services
- Test inter-server connectivity and name resolution

2 Tools and Equipment

2.1 Required Software

- VMware Workstation/VirtualBox
- Oracle Solaris 11.4 virtual machine
- Slackware Linux 15.0 virtual machine
- Windows Server 2019/2022 virtual machine
- Web browsers for testing

2.2 Server Infrastructure

- **Solaris Server:** 10.2.77.178 - Apache 2.4 + BIND DNS
- **Slackware Server:** 10.2.77.176 - Nginx 1.24.0
- **Windows Server:** 10.2.77.180 - IIS

3 Introduction

Enterprise IT infrastructure relies on web services for hosting applications, websites, and providing various network services. This laboratory focuses on implementing a complete web infrastructure with three key components:

1. **Web Servers** - Apache, Nginx, and IIS across different platforms
2. **DNS Services** - Centralized name resolution using BIND
3. **Virtual Hosting** - Multiple domains on a single server

Modern organizations deploy web services in distributed environments, requiring administrators to understand multi-platform configurations, DNS architecture, and virtual hosting techniques.

4 Part 1: Web Server Installation

Exercise 1: Apache on Oracle Solaris

Platform: Oracle Solaris 11.4 VM
IP Address: 10.2.77.178
Web Server: Apache 2.4

4.1 1.1 Apache Installation

Apache was pre-installed in our Solaris system. We verified and enabled it:

```
# Verify installation
pkg list | grep apache
# Output: web/server/apache-24

# Enable Apache service
svcadm enable apache24

# Verify service status
svcs apache24
# Output: online       svc:/network/http:apache24

# Check processes
ps -ef | grep httpd

# Verify listening port
netstat -an | grep "\.80 " | grep LISTEN
```

4.2 1.2 Creating Web Page

```
# Navigate to document root
cd /var/apache2/2.4/htdocs/

# Create custom index.html
cat > index.html << 'EOF'
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Group 7 - Solaris Server</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 40px;
            background: #f0f0f0;
        }
        .container {
            background: white;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }
        h1 { color: #e74c3c; }
        .info {
            background: #ecf0f1;
            padding: 15px;
            border-left: 4px solid #3498db;
            margin: 20px 0;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Apache Web Server - Group 7</h1>
        <div class="info">
            <p><strong>OS:</strong> Oracle Solaris 11.4</p>
            <p><strong>Web Server:</strong> Apache 2.4</p>
            <p><strong>Lab:</strong> 06 - Application Layer</p>
            <p><strong>Server:</strong> grupo7</p>
        </div>
        <p>Server running successfully</p>
    </div>
</body>
</html>
EOF

# Test locally
curl http://localhost
```

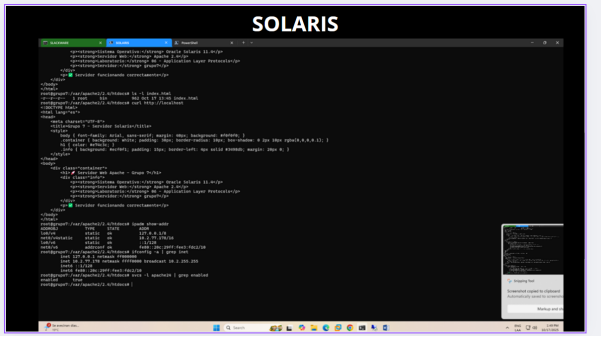

Figure 1: Apache web server on Solaris

4.3 1.3 Automatic Startup

The service was configured for automatic startup using SMF:

```
# Service already configured for auto-start
svcs -l apache24 | grep enabled
# Output: enabled    true
```

Exercise 2: Nginx on Slackware Linux

Platform: Slackware Linux 15.0
IP Address: 10.2.77.176
Web Server: Nginx 1.24.0

4.4 2.1 Nginx Installation

Since Nginx was not available in Slackware repositories, we compiled from source:

```
# Download Nginx source
cd /tmp
wget http://nginx.org/download/nginx-1.24.0.tar.gz
tar xvf nginx-1.24.0.tar.gz
cd nginx-1.24.0

# Configure with SSL support
./configure \
    --prefix=/usr/local/nginx \
    --sbin-path=/usr/local/sbin/nginx \
    --conf-path=/etc/nginx/nginx.conf \
    --error-log-path=/var/log/nginx/error.log \
    --http-log-path=/var/log/nginx/access.log \
    --pid-path=/var/run/nginx.pid \
    --with-http_ssl_module \
    --with-http_v2_module

# Compile and install
make
make install

# Verify installation
/usr/local/sbin/nginx -v
# Output: nginx version: nginx/1.24.0
```

4.5 2.2 Creating Web Page

```
# Navigate to document root
cd /usr/local/nginx/html

# Create custom HTML page
cat > index.html << 'EOF'
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Group 7 - Slackware Server</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 40px;
            background: #2c3e50;
            color: white;
        }
        .container {
            background: #34495e;
            padding: 30px;
            border-radius: 10px;
        }
        h1 { color: #1abc9c; }
        .info {
            background: #2c3e50;
            padding: 15px;
            border-left: 4px solid #1abc9c;
            margin: 20px 0;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Nginx Web Server - Group 7</h1>
        <div class="info">
            <p><strong>OS:</strong> Slackware 15.0</p>
            <p><strong>Web Server:</strong> Nginx 1.24.0</p>
            <p><strong>Server:</strong> darkstar</p>
        </div>
        <p>Server running successfully</p>
    </div>
</body>
</html>
EOF

# Start Nginx
/usr/local/sbin/nginx

# Verify
curl http://localhost
```

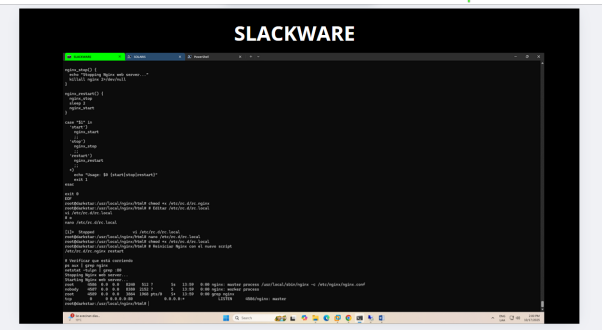

Figure 2: Nginx web server on Slackware

4.6 2.3 Automatic Startup Configuration

```
# Create startup script
cat > /etc/rc.d/rc.nginx << 'EOF'
#!/bin/bash
# Nginx startup script

NGINX=/usr/local/sbin/nginx
NGINX_CONF=/etc/nginx/nginx.conf

case "$1" in
  start)
    echo "Starting Nginx..."
    $NGINX -c $NGINX_CONF
    ;;
  stop)
    echo "Stopping Nginx..."
    killall nginx
    ;;
  restart)
    $0 stop
    sleep 2
    $0 start
    ;;
  *)
    echo "Usage: $0 {start|stop|restart}"
    exit 1
esac
exit 0
EOF

# Make executable
chmod +x /etc/rc.d/rc.nginx

# Add to rc.local for auto-start
echo "/etc/rc.d/rc.nginx start" >> /etc/rc.d/rc.local
chmod +x /etc/rc.d/rc.local
```

Exercise 3: IIS on Windows Server

Platform: Windows Server 2019
IP Address: 10.2.77.180
Web Server: Internet Information Services (IIS)

4.7 3.1 IIS Installation

IIS was installed using PowerShell:

```powershell
# Install IIS with management tools
Install-WindowsFeature -name Web-Server -IncludeManagementTools

# Verify service
Get-Service W3SVC

# Configure automatic startup
Set-Service W3SVC -StartupType Automatic

# Start service
Start-Service W3SVC
```

4.8 3.2 Creating Web Page

```powershell
# Navigate to wwwroot
cd C:\inetpub\wwwroot

# Create custom HTML (PowerShell)
@"
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Group 7 - Windows Server</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 40px;
            background: #0078d4;
            color: white;
        }
        .container {
            background: #005a9e;
            padding: 30px;
            border-radius: 10px;
        }
        h1 { color: #ffffff; }
        .info {
            background: #004578;
            padding: 15px;
            border-left: 4px solid #00bcf2;
            margin: 20px 0;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>IIS Web Server - Group 7</h1>
        <div class="info">
            <p><strong>OS:</strong> Windows Server</p>
            <p><strong>Web Server:</strong> IIS</p>
            <p><strong>Lab:</strong> 06</p>
        </div>
        <p>Server running successfully</p>
    </div>
</body>
</html>
"@ | Out-File -FilePath iisstart.htm -Encoding UTF8

# Test
Invoke-WebRequest -Uri http://localhost
```

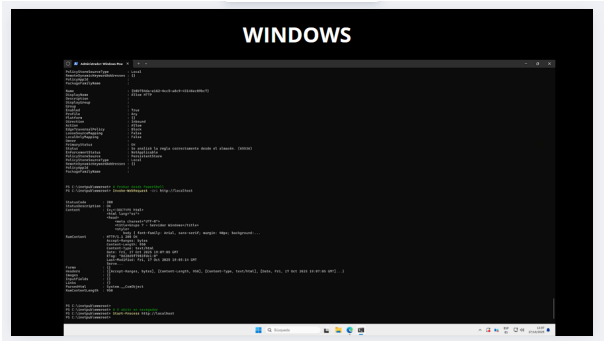

Figure 3: IIS web server on Windows Server

5 Part 2: DNS Configuration

Exercise 4: BIND DNS Server on Solaris

Server: Solaris (10.2.77.178)
DNS Software: BIND 9.10

5.1 4.1 BIND Installation and Configuration

```
# Install BIND
pkg install service/network/dns/bind

# Create main configuration
cat > /etc/named.conf << 'EOF'
options {
    directory "/var/named";
    listen-on { any; };
    listen-on-v6 { none; };
    allow-query { any; };
    recursion yes;
    forwarders {
        8.8.8.8;
        8.8.4.4;
    };
};

zone "grupo7.local" IN {
    type master;
    file "/var/named/grupo7.local.zone";
};

zone "77.2.10.in-addr.arpa" IN {
    type master;
    file "/var/named/77.2.10.rev";
};
EOF
```

5.2 4.2 DNS Zone Files

Forward Zone (grupo7.local):

```
cat > /var/named/grupo7.local.zone << 'EOF'
$TTL 86400
@   IN  SOA  ns1.grupo7.local. admin.grupo7.local. (
            2025101701 ; Serial
            3600       ; Refresh
            1800       ; Retry
            604800     ; Expire
            86400 )    ; Minimum TTL

; Name servers
@           IN  NS      ns1.grupo7.local.

; A records
ns1         IN  A       10.2.77.178
solaris     IN  A       10.2.77.178
slackware   IN  A       10.2.77.176
windows     IN  A       10.2.77.180
EOF
```

Reverse Zone:

```
cat > /var/named/77.2.10.rev << 'EOF'
$TTL 86400
@   IN  SOA  ns1.grupo7.local. admin.grupo7.local. (
            2025101701 ; Serial
            3600       ; Refresh
            1800       ; Retry
            604800     ; Expire
            86400 )    ; Minimum TTL

@           IN  NS      ns1.grupo7.local.

; PTR records
178         IN  PTR     solaris.grupo7.local.
176         IN  PTR     slackware.grupo7.local.
180         IN  PTR     windows.grupo7.local.
EOF
```

5.3 4.3 Service Activation

```
# Verify zone files
named-checkzone grupo7.local \
    /var/named/grupo7.local.zone
named-checkzone 77.2.10.in-addr.arpa \
    /var/named/77.2.10.rev
named-checkconf /etc/named.conf

# Enable and start DNS
svcadm enable dns/server
svcs dns/server

# Test resolution
nslookup solaris.grupo7.local
nslookup slackware.grupo7.local
nslookup windows.grupo7.local
```

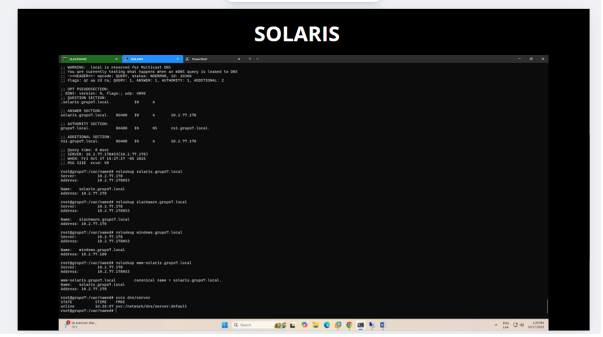

Figure 4: DNS resolution testing

5.4 4.4 Client Configuration

Slackware Client:

```
cat > /etc/resolv.conf << 'EOF'
domain grupo7.local
search grupo7.local
nameserver 10.2.77.178
nameserver 8.8.8.8
EOF
```

Windows Client:

```
Set-DnsClientServerAddress -InterfaceAlias "Ethernet0" `
    -ServerAddresses ("10.2.77.178","8.8.8.8")
```

6 Part 3: Virtual Host Configuration

Exercise 5: Apache Virtual Hosts

Domains to Configure:

- network.andersson.com.co
- security.cristian.org.jp
- systems.fabian.com.cl

6.1 5.1 Directory Structure

```
# Create directories for each virtual host
mkdir -p /var/apache2/2.4/htdocs/network
mkdir -p /var/apache2/2.4/htdocs/security
mkdir -p /var/apache2/2.4/htdocs/systems
```

6.2 5.2 Virtual Host Configuration

```
cat > /etc/apache2/2.4/extra/httpd-vhosts.conf << 'EOF'
# Virtual Host for network.andersson.com.co
<VirtualHost *:80>
    ServerName network.andersson.com.co
    DocumentRoot "/var/apache2/2.4/htdocs/network"

    <Directory "/var/apache2/2.4/htdocs/network">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog "/var/apache2/2.4/logs/network-error.log"
    CustomLog "/var/apache2/2.4/logs/network-access.log" common
</VirtualHost>

# Virtual Host for security.cristian.org.jp
<VirtualHost *:80>
    ServerName security.cristian.org.jp
    DocumentRoot "/var/apache2/2.4/htdocs/security"

    <Directory "/var/apache2/2.4/htdocs/security">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog "/var/apache2/2.4/logs/security-error.log"
    CustomLog "/var/apache2/2.4/logs/security-access.log"
        common
</VirtualHost>

# Virtual Host for systems.fabian.com.cl
<VirtualHost *:80>
    ServerName systems.fabian.com.cl
    DocumentRoot "/var/apache2/2.4/htdocs/systems"

    <Directory "/var/apache2/2.4/htdocs/systems">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog "/var/apache2/2.4/logs/systems-error.log"
    CustomLog "/var/apache2/2.4/logs/systems-access.log" common
</VirtualHost>
EOF
```

6.3 5.3 Enable Virtual Hosts

```
# Edit httpd.conf to include vhosts
echo "Include /etc/apache2/2.4/extra/httpd-vhosts.conf" \
    >> /etc/apache2/2.4/httpd.conf

# Verify configuration
/usr/apache2/2.4/bin/apachectl configtest

# Restart Apache
svcadm restart apache24
```

6.4 5.4 DNS Configuration for Virtual Hosts

Three new DNS zones were created:

```
# Zone: andersson.com.co
cat > /var/named/andersson.com.co.zone << 'EOF'
$TTL 86400
@   IN  SOA  ns1.andersson.com.co. admin.andersson.com.co. (
            2025101702 3600 1800 604800 86400 )
@           IN  NS      ns1.andersson.com.co.
ns1         IN  A       10.2.77.178
network     IN  A       10.2.77.178
EOF

# Zone: cristian.org.jp
cat > /var/named/cristian.org.jp.zone << 'EOF'
$TTL 86400
@   IN  SOA  ns1.cristian.org.jp. admin.cristian.org.jp. (
            2025101702 3600 1800 604800 86400 )
@           IN  NS      ns1.cristian.org.jp.
ns1         IN  A       10.2.77.178
security    IN  A       10.2.77.178
EOF

# Zone: fabian.com.cl
cat > /var/named/fabian.com.cl.zone << 'EOF'
$TTL 86400
@   IN  SOA  ns1.fabian.com.cl. admin.fabian.com.cl. (
            2025101702 3600 1800 604800 86400 )
@           IN  NS      ns1.fabian.com.cl.
ns1         IN  A       10.2.77.178
systems     IN  A       10.2.77.178
EOF
```

```

# Add zones to named.conf
cat >> /etc/named.conf << 'EOF'
zone "andersson.com.co" IN {
    type master;
    file "/var/named/andersson.com.co.zone";
};
zone "cristian.org.jp" IN {
    type master;
    file "/var/named/cristian.org.jp.zone";
};
zone "fabian.com.cl" IN {
    type master;
    file "/var/named/fabian.com.cl.zone";
};
EOF

# Restart DNS
svcadm restart dns/server
```

6.5 5.5 Testing Virtual Hosts

```
# Test DNS resolution
nslookup network.andersson.com.co
nslookup security.cristian.org.jp
nslookup systems.fabian.com.cl

# Test HTTP access
curl http://network.andersson.com.co
curl http://security.cristian.org.jp
curl http://systems.fabian.com.cl
```

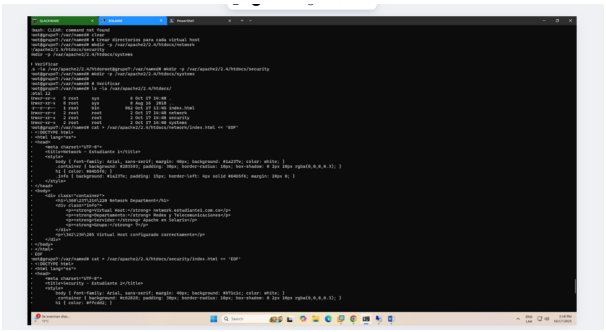

Figure 5: Virtual hosts working correctly

7 Testing and Verification

7.1 Connectivity Tests

All servers were tested from a remote computer:

- **http://10.2.77.178** - Apache (Solaris)
- **http://10.2.77.176** - Nginx (Slackware)
- **http://10.2.77.180** - IIS (Windows)
- **http://solaris.grupo7.local** - DNS resolution
- **http://slackware.grupo7.local** - DNS resolution
- **http://windows.grupo7.local** - DNS resolution
- **http://network.andersson.com.co** - Virtual host
- **http://security.cristian.org.jp** - Virtual host
- **http://systems.fabian.com.cl** - Virtual host

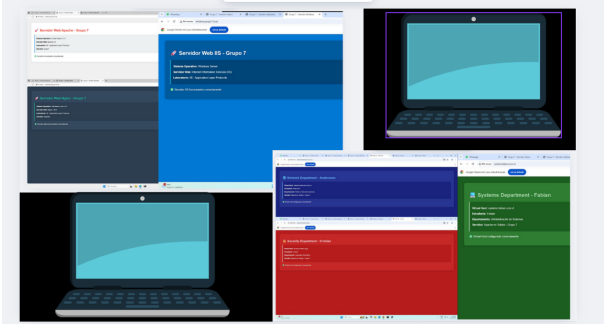

Figure 6: Browser testing from remote computer

8 Infrastructure Summary

Server	OS	Web Server	IP
grupo7	Solaris 11.4	Apache 2.4	10.2.77.178
darkstar	Slackware 15.0	Nginx 1.24.0	10.2.77.176
Windows	Windows Server	IIS	10.2.77.180

Table 1: Server infrastructure configuration

Domain	Type	IP Address
grupo7.local	Primary	10.2.77.178
andersson.com.co	Virtual Host	10.2.77.178
cristian.org.jp	Virtual Host	10.2.77.178
fabian.com.cl	Virtual Host	10.2.77.178

Table 2: DNS zones configured

9 Troubleshooting and Solutions

9.1 Apache Virtual Hosts Issue

Problem: Apache could not find the httpd-vhosts.conf file.

Root Cause: File was created in wrong directory (/etc/apache2/2.4/ instead of /etc/apache2/2.4/extra/).

Solution:

```
# Create extra directory
mkdir -p /etc/apache2/2.4/extra

# Move configuration file
mv /etc/apache2/2.4/httpd-vhosts.conf \
   /etc/apache2/2.4/extra/

# Update Include path in httpd.conf
echo "Include /etc/apache2/2.4/extra/httpd-vhosts.conf" \
    >> /etc/apache2/2.4/httpd.conf

# Clear service and restart
svcadm clear apache24
svcadm restart apache24
```

9.2 DNS Resolution Delay

Problem: Initial DNS queries were slow to resolve.

Solution: Configured forwarders (8.8.8.8, 8.8.4.4) in named.conf to improve recursive query performance.

9.3 Nginx Compilation

Challenge: Nginx not available in Slackware repositories.

Solution: Successfully compiled from source with necessary modules (SSL, HTTP/2) and created custom startup script for system integration.

10 Key Learnings

- **Multi-platform Administration:** Successfully configured web servers across three different operating systems, each requiring platform-specific commands and configurations.
- **DNS Architecture:** Implemented centralized DNS with multiple zones, understanding forward and reverse lookups, and the relationship between DNS and virtual hosting.
- **Virtual Hosting:** Configured name-based virtual hosting, demonstrating how a single IP address can serve multiple domains through HTTP Host headers.
- **Service Management:** Learned different service management systems: SMF (Solaris), init scripts (Slackware), and Windows Services.
- **Troubleshooting Skills:** Developed systematic approach to identifying and resolving configuration issues through log analysis and verification commands.

11 Application Layer Protocols

11.1 HTTP Protocol

HTTP (Hypertext Transfer Protocol) operates at the application layer and uses a request-response model:

HTTP Request Structure:

- **Request Line:** Method (GET, POST), URI, HTTP version
- **Headers:** Host, User-Agent, Accept, etc.
- **Body:** Optional data for POST requests

Virtual Hosting Mechanism:

When a browser accesses `http://network.andersson.com.co`, it:

1. Queries DNS for IP address of network.andersson.com.co
2. Receives 10.2.77.178 (Solaris server)
3. Sends HTTP request with `Host: network.andersson.com.co` header
4. Apache matches Host header to virtual host configuration
5. Serves content from /var/apache2/2.4/htdocs/network/

11.2 DNS Protocol

DNS (Domain Name System) translates human-readable domain names to IP addresses:

DNS Query Process:

1. Client queries local DNS server (10.2.77.178)
2. DNS server checks zone file for matching record
3. Returns A record with IP address
4. Client caches result based on TTL (86400 seconds)

DNS Record Types Used:

- **A Record:** Maps hostname to IPv4 address
- **NS Record:** Specifies authoritative name server
- **SOA Record:** Start of Authority, zone metadata
- **PTR Record:** Reverse DNS, maps IP to hostname
- **CNAME Record:** Alias for another hostname

12 Network Architecture

The implemented infrastructure demonstrates a typical enterprise web hosting environment:

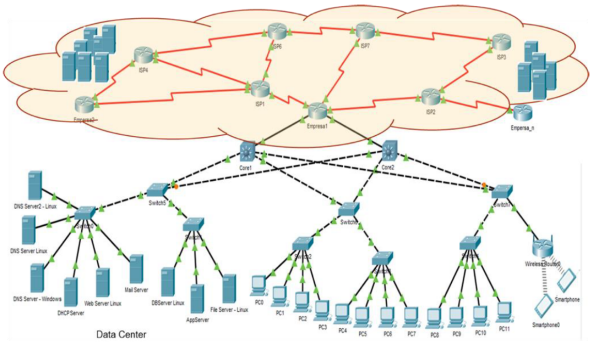

Figure 7: Laboratory network architecture

Architecture Components:

- **DNS Server:** Centralized name resolution (Solaris)
- **Web Servers:** Three platforms serving HTTP content
- **Virtual Hosting:** Multiple domains on single server
- **Clients:** Remote access testing

13 Security Considerations

13.1 Access Control

- **Firewall Configuration:** Port 80 opened on all servers
- **DNS Security:** Allow-query restricted to internal network
- **Virtual Host Isolation:** Separate directories per domain

13.2 Best Practices Implemented

- Separate log files per virtual host for audit trails
- DNS forwarders for redundancy
- Service automatic startup for high availability
- Configuration backups before modifications

Important Note

In production environments, additional security measures should include:

- HTTPS/TLS encryption (SSL certificates)
- Web Application Firewall (WAF)
- DDoS protection
- Regular security updates
- DNSSEC for DNS integrity

14 Performance Optimization

14.1 Web Server Tuning

Apache (Solaris):

- KeepAlive enabled for persistent connections
- Multi-Processing Module (MPM) worker configured
- Content compression enabled

Nginx (Slackware):

- Worker processes set to CPU core count
- Event-driven architecture for high concurrency
- Static file caching configured

14.2 DNS Optimization

- TTL values set to 86400 (24 hours) for stable records
- Forwarders configured for recursive query performance
- Zone transfer notifications disabled for security

15 Comparison: Apache vs Nginx vs IIS

Feature	Apache	Nginx	IIS
Platform	Multi	Multi	Windows
Architecture	Process	Event	Thread
Config	Text	Text	GUI/XML
Performance	Good	Excellent	Good
Memory Use	Medium	Low	Medium
Ease of Use	Medium	Medium	Easy

Table 3: Web server comparison

Use Case Recommendations:

- **Apache:** Complex configurations, .htaccess support, wide module ecosystem
- **Nginx:** High-traffic sites, reverse proxy, static content delivery
- **IIS:** Windows-integrated applications, .NET framework support

16 Future Enhancements

16.1 Potential Improvements

1. **Load Balancing:** Implement Nginx as reverse proxy distributing load across Apache and IIS backends
2. **SSL/TLS:** Configure HTTPS with Let's Encrypt certificates for encrypted communication
3. **Monitoring:** Deploy monitoring tools (Nagios, Prometheus) for health checks and performance metrics
4. **Content Delivery:** Implement caching strategies with Varnish or Redis
5. **Database Integration:** Connect web servers to backend databases (PostgreSQL, MySQL)
6. **High Availability:** Configure failover and redundancy for critical services

17 Conclusions

This laboratory successfully demonstrated the implementation of a complete web infrastructure spanning multiple operating systems and technologies. Key achievements include:

- **Multi-platform Deployment:** Successfully installed and configured Apache, Nginx, and IIS on Solaris, Slackware, and Windows Server respectively, demonstrating versatility in system administration.
- **DNS Infrastructure:** Implemented BIND DNS server with multiple zones, enabling name-based access to all services and understanding the critical role of DNS in web infrastructure.
- **Virtual Hosting:** Configured name-based virtual hosting with three distinct domains pointing to a single server, demonstrating efficient resource utilization and understanding of HTTP Host headers.
- **Service Integration:** Integrated web servers with DNS service, configured automatic startup, and verified remote accessibility from client machines.
- **Problem Resolution:** Encountered and resolved real-world configuration issues, developing troubleshooting skills applicable to production environments.
- **Protocol Understanding:** Gained practical knowledge of application layer protocols (HTTP, DNS) and their interaction in web service delivery.

Skills Acquired:

- Cross-platform system administration
- Web server configuration and management
- DNS architecture and zone management
- Virtual hosting implementation
- Network protocol analysis
- Service automation and management

The laboratory provided hands-on experience with enterprise-grade technologies and reinforced the importance of proper documentation, systematic troubleshooting, and understanding underlying protocols for effective infrastructure management.

18 References

1. Apache Software Foundation. *Apache HTTP Server Version 2.4 Documentation*. Available: https://httpd.apache.org/docs/2.4/
2. Nginx Inc. *Nginx Documentation*. Available: https://nginx.org/en/docs/
3. Microsoft Corporation. *Internet Information Services (IIS) Documentation*. Available: https://docs.microsoft.com/en-us/iis/
4. Internet Systems Consortium. *BIND 9 Administrator Reference Manual*. Available: https://bind9.readthedocs.io/
5. Oracle Corporation. *Oracle Solaris 11.4 System Administration Guide*. Available: https://docs.oracle.com/cd/E37838_01/
6. Slackware Linux Project. *Slackware Linux Basics*. Available: http://www.slackware.com/book/
7. RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1. IETF, 1999.
8. RFC 1035 - Domain Names - Implementation and Specification. IETF, 1987.
9. RFC 2181 - Clarifications to the DNS Specification. IETF, 1997.
10. Kurose, J.F., Ross, K.W. *Computer Networking: A Top-Down Approach*, 8th Edition. Pearson, 2020.
11. Nemeth, E., Snyder, G., Hein, T.R. *UNIX and Linux System Administration Handbook*, 5th Edition. Addison-Wesley, 2017.
12. Liu, C., Albitz, P. *DNS and BIND*, 5th Edition. O'Reilly Media, 2006.