Solution Architecture for the Given Assignment

Given the requirement of designing a solution that involves five distinct systems (S1, S2, S3, S4, and S5), let's go step by step and address each part of the assignment in detail, ensuring that the solution is both scalable and fault-tolerant.

Problem Understanding and Requirements Recap:

1. **Systems Involved:**
    - **S1**: Linux-based standalone app (MySQL database), in a local network, sharing data with S2.
    - **S2**: PHP/MySQL Web app, in local/public network, receiving data from S1.
    - **S3**: PHP/PostgreSQL Web app, in local network, sharing data with S5.
    - **S4**: Python/Django/SQLite3 Web app, in local network, sharing data with S5.
    - **S5**: Java Web app, in local/public network, receiving data from S3/S4.

2. **System Requirements:**
    - **24x7 Availability**: All systems need to be available round the clock.
    - **End-User Access**: Users need to access all systems via a public network.
    - **Data Backup**: Weekly backup for all systems.

3. **Infrastructure Details:**
    - **2 Physical Servers Available** in scenario 1.
    - **No limit on physical servers** and **2 Public Network Connections** in scenario 2.

Solution 1: Two Physical Servers Available and One Public Network Connection

In this scenario, we are constrained by the number of physical servers but have access to only one public network connection. Given these constraints, we will try to design a solution that maximizes system performance while ensuring availability and fault tolerance.

1. Physical Server Plan

- **Server 1** (Primary Server): Host a combination of systems that are in the local network and share data among each other.
    - Host **S1** (Linux-based standalone app with MySQL).
    - Host **S2** (PHP/MySQL Web app).
    - Host **S3** (PHP/PostgreSQL Web app).
    - **S3** can be hosted on Server 1 because it's part of the same local network and communicates with S5, reducing the need for separate infrastructure.
    -
- **Server 2** (Secondary Server): Host the remaining systems.
    - Host **S4** (Python/Django/SQLite3 Web app).
    - Host **S5** (Java Web app).

2. Systems Architecture

The architecture needs to consider data flow and communication between systems:

- **S1** sends data to **S2** (MySQL to MySQL communication).
- **S3** and **S4** send data to **S5** (PHP/PostgreSQL and Python/Django to Java Web app).
- **S5** acts as a centralized platform for the external communication with users (public network).

3. Network Design

We can leverage **private network zones** for communication between systems within the local network and a **DMZ (Demilitarized Zone)** to expose the systems to the public network. The public-facing services like S2 and S5 should be exposed to the public network through a **reverse proxy** (e.g., **NGINX** or **HAProxy**).

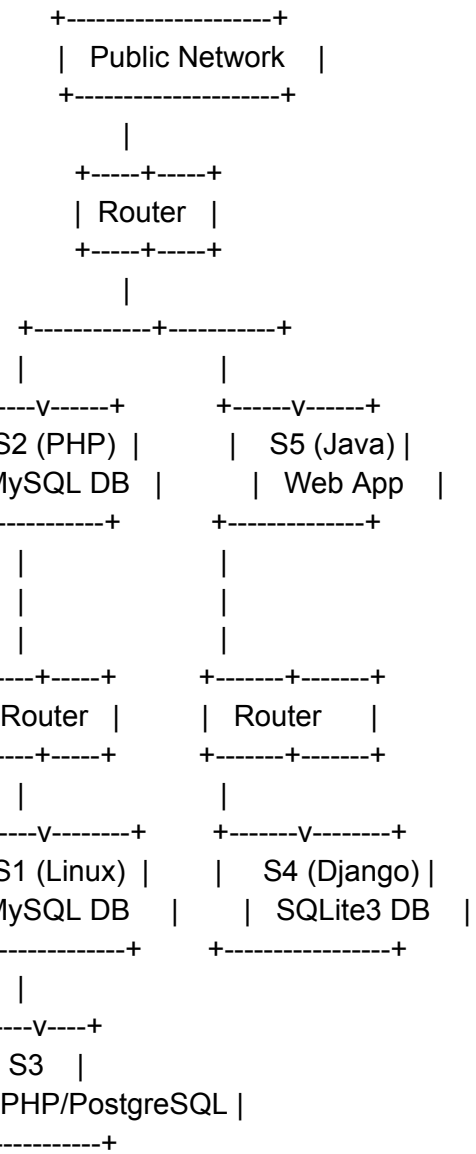
Design Overview:

- **Private Network Zone**: Systems like S1, S3, and S4 are isolated in a private network to communicate without public access.
- **Public Network Zone**: Systems like S2 and S5 are in the public network zone for access from the internet.

Network Flow:

1. Internal communication is done through the private network (e.g., **192.168.x.x**).
2. External communication for S2 and S5 is routed through the public network.
3. Use **firewalls** and **NAT** (Network Address Translation) for additional security.

4. Deployment Diagram



5. Backup Strategy:

- **Backup Tools**: Use **rsync**, **pg_dump** for PostgreSQL, and **MySQL dump** for MySQL backups.
- **Frequency**: Weekly backup scheduled using **cron jobs**.
- Store backups on a remote server or a cloud-based storage service (e.g., **AWS S3** or **Google Cloud Storage**) for redundancy.

Solution 2: Unlimited Physical Servers and Two Public Network Connections Available

In this scenario, we can make full use of the available resources and create a more scalable, fault-tolerant architecture. We'll split the systems across multiple servers, and optimize for availability, performance, and security.

1. Physical Server Plan

- **Server 1**: Host **S1** (Linux/MySQL) and **S2** (PHP/MySQL).
- **Server 2**: Host **S3** (PHP/PostgreSQL) and **S4** (Python/Django/SQLite).
- **Server 3**: Host **S5** (Java Web app).

2. Systems Architecture

In this design, systems are isolated by their functionality but will continue sharing data as per the requirements:

- **S1** communicates with **S2**.
- **S3** and **S4** share data with **S5**.
- **S2** and **S5** are exposed to the public network through load balancers.

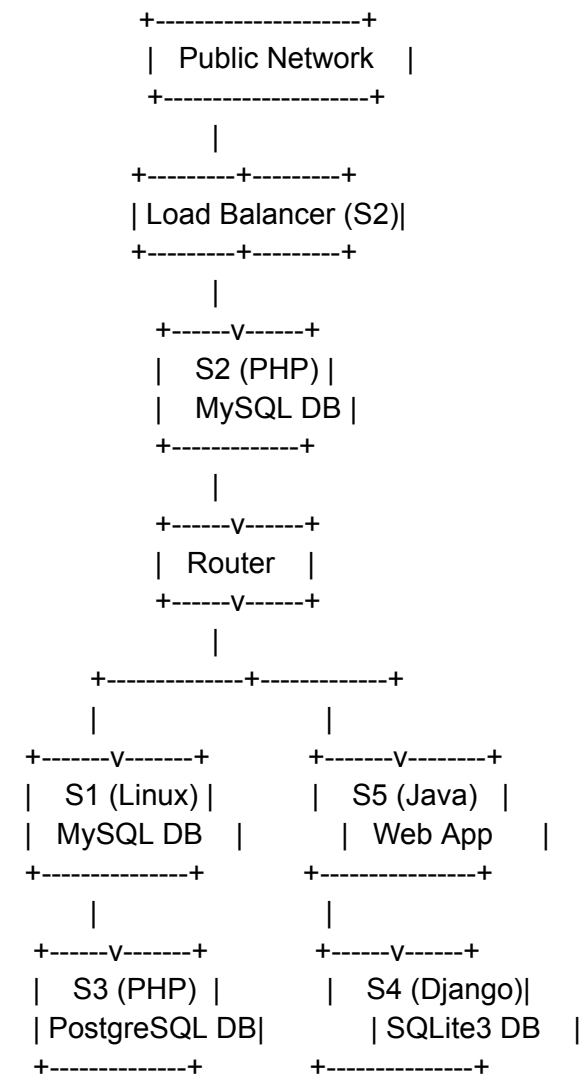
3. Network Design

- **Internal Communication**: Systems are distributed across different servers but still in the same private network. Private IPs are used for internal communication.
- **Load Balancers**: Use **HAProxy** or **NGINX** to distribute traffic across multiple instances of public-facing systems (**S2**, **S5**).
- **Public Network Access**: **Two Public Network Connections** allow redundancy and improved availability.

Network Design:

1. Systems S2 and S5 should be fronted by a **load balancer** to handle incoming user traffic.
2. **Firewalls** and **VPN** should be used to protect internal communication.

4. Deployment Diagram



5. Backup Strategy:

Use the same backup strategy as described in Solution 1, but with an added redundancy layer:

- Implement **incremental backups** in addition to full backups for efficient data management.
- Store backups in geographically distributed locations to prevent data loss.

Conclusion

Why this Solution?

- **Fault Tolerance**: Multiple servers (in Solution 2) and load balancing ensure that the system can continue operating even if one server fails.
- **Security**: The use of firewalls, VPNs, and network segmentation protects internal systems while exposing only necessary systems (S2, S5) to the public.
- **Scalability**: The solution is designed to scale by adding additional servers if needed, especially in the case of increased load on public-facing systems.
- **Backup & Data Integrity**: Automated weekly backups ensure that data is protected, and redundancy in backup storage minimizes risk of data loss.