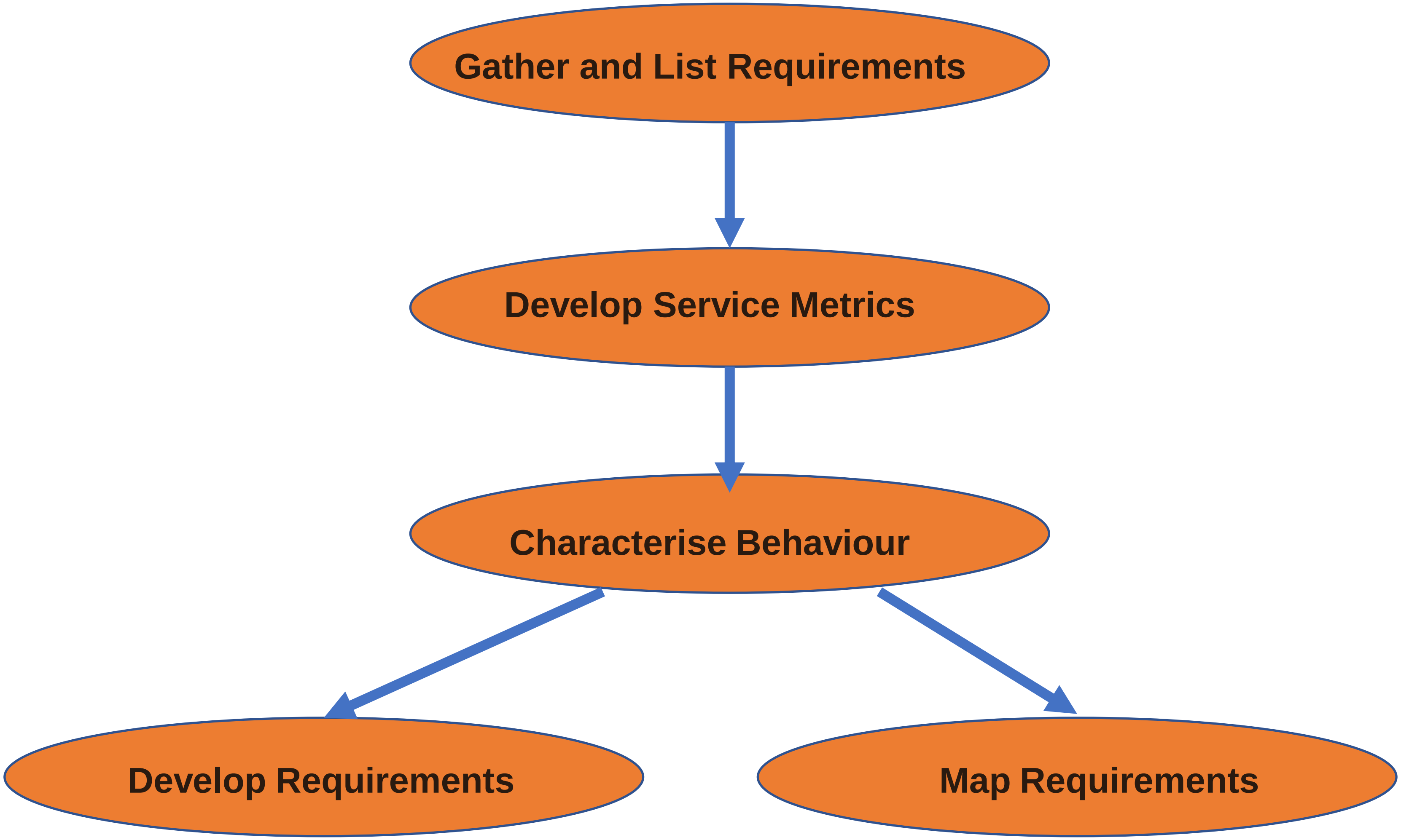
UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Week 5b Network Architectures

CSIT985

Strategic Network Design

Requirement Analysis Process



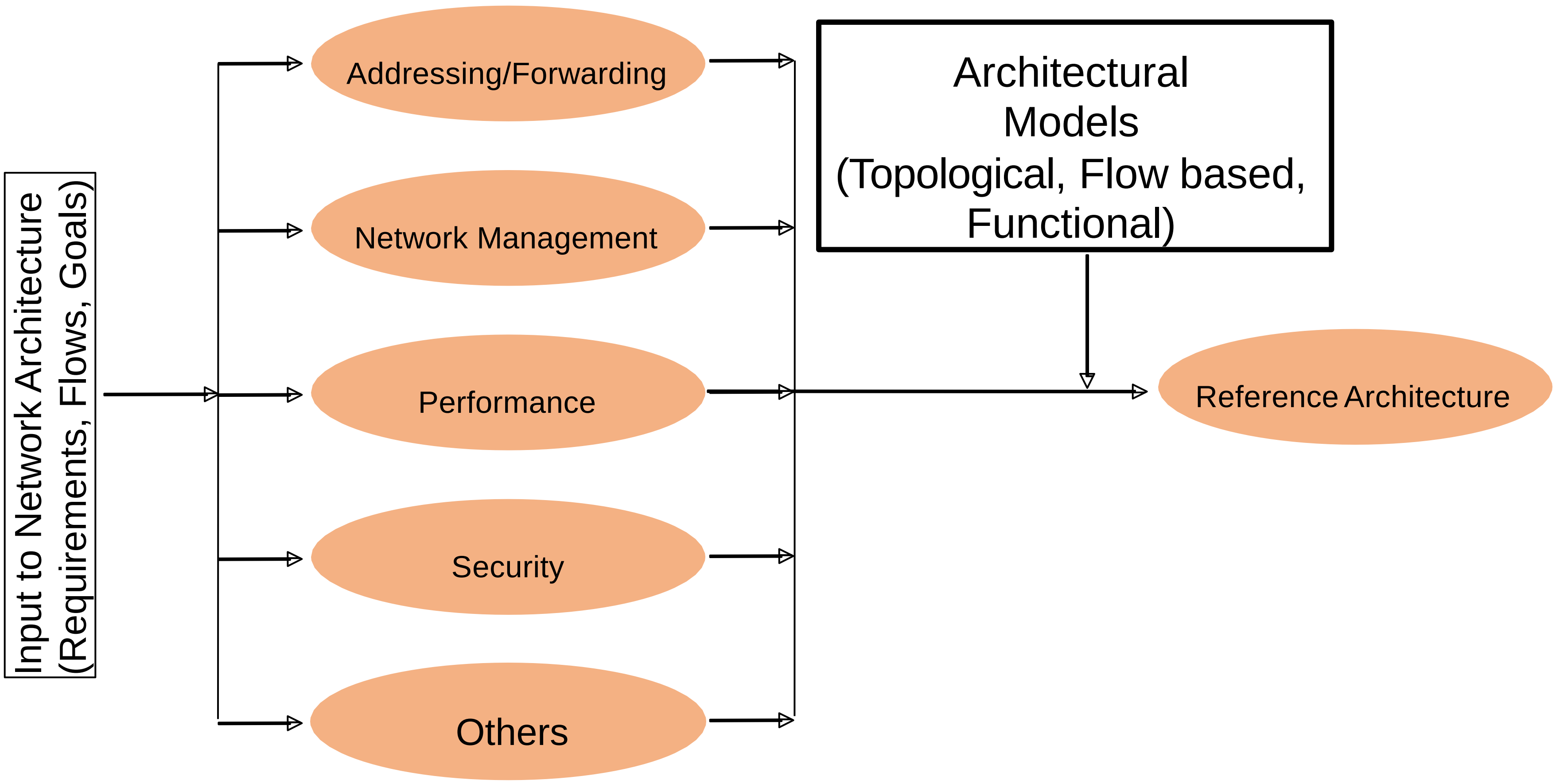
A useful quote from McCabe

'…Network architecture and design are attempts to solve nonlinear problems, and figuring out where to begin can be difficult. You cannot start at the top without some understanding of the capabilities of the individual components, and you cannot easily pick components until you understand the top-down requirements…'

Another useful quote

'…Network architecture and design development is no longer simple enough that tricks will work: it must be done in a systematic and reproducible manner. Even if the complex network architecture/design is "tricked" into existence, it cannot be maintained. Smart customers are beyond the stage at which they would hire a wizard to work magic…' (p. 213).

Process Model for Component Architecture Approach



Component Architectures (Based on Network Functions)

UNIVERSITY OF WOLLONGONG AUSTRALIA

Background

Section One

Analysis

Requirements, Flows, Risk

Section Two

Architecture

Relationships within and between Network Functions

Section Three

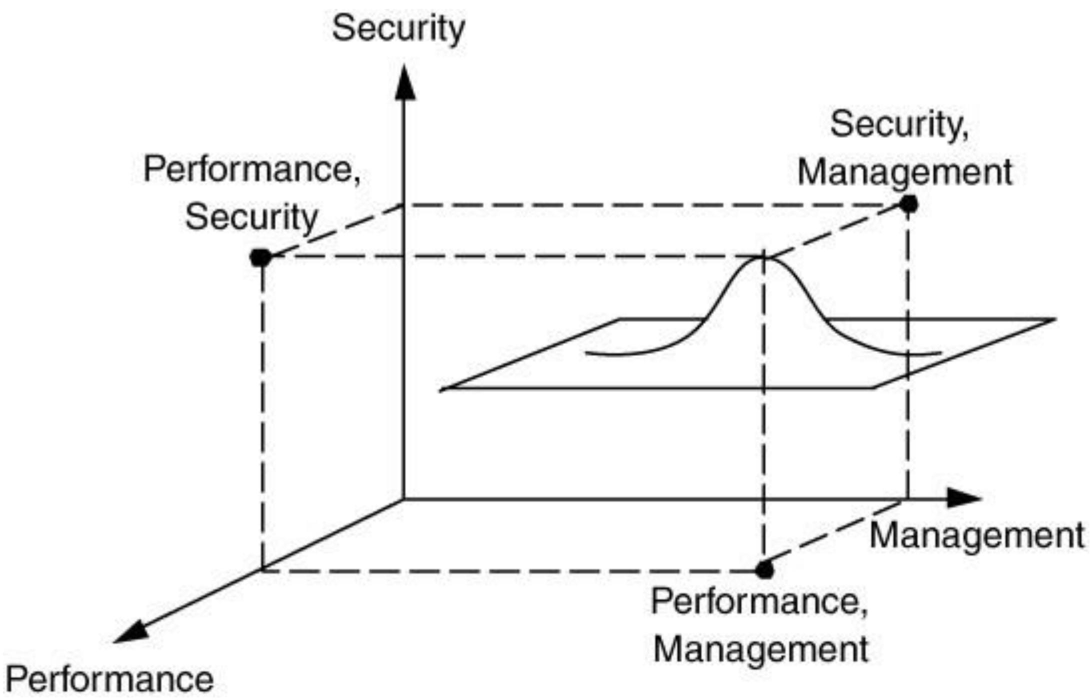
Design

Technology, Equipment Choices, Connectivity Choices

Information Flows

Architecture and Design I

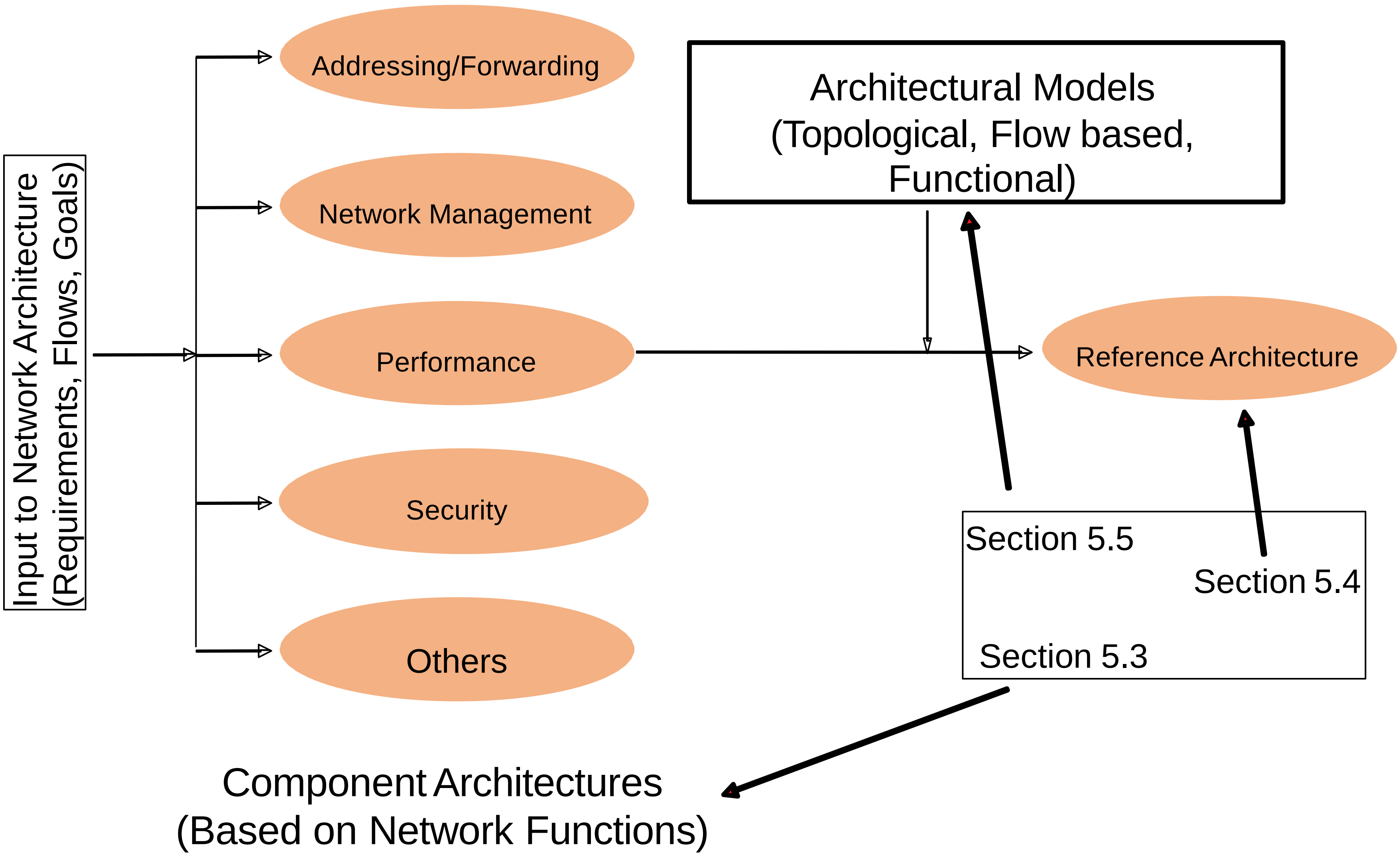
- The aim is to solve multi-dimensional problems based on systematic analysis



5.2.1 Architecture v Design

Architecture		Design
Broad	← Scope →	Focused
Generalised	← Level of Detail →	In Depth
Relationships	← Description →	Technologies
Independent	← Location →	Dependent

Process Model for Component Architecture Approach



Component Architectures

- Addressing and routing
- Network Management
- Performance
- Security
- Other

Component Architectures

Are made up of

① functions

② mechanisms

③ internal relationships

Component Architectures

- Functions
  - Major capability of network
- Mechanisms
  - Hardware and software that help a network achieve each capability

Functions	Description of capability	Example subset of mechanism
Addressing and Routing	Provides robust and flexible connectivity between devices	• **Addressing**: ways to allocate and aggregate address space • **Routing**: routers, routing protocols, ways to manipulate routing flows

Component Architectures

Functions	Description of capability	Example subset of mechanism
Network Management	Provides monitoring, configuring and troubleshooting for the network	• Network Management (NM) protocols • NM devices • Ways to configure NM in the network

Component Architectures

Functions	Description of capability	Example subset of mechanism
Performance	Provides network resources to support requirements for capacity, delay and RMA	• QoS (quality of service) • SLA (Service Level Agreement) • Policies

Component Architectures

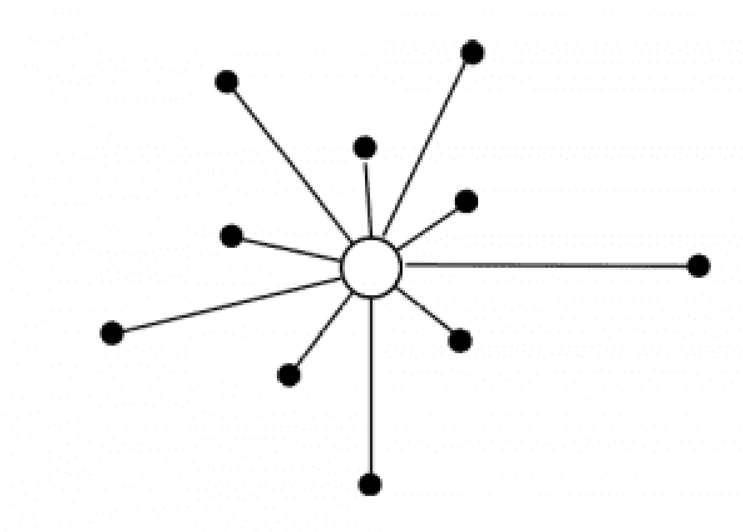
Functions	Description of capability	Example subset of mechanism
Security	Restricts unauthorised access, usage and visibility within networks to reduce the threat and effects of attacks	• Firewalls • Security policies and procedures • Filters and Access Control lists

Component Architectures

- Internal Relationships
  - Trade-offs and Dependencies
    - Trade-offs
      - Decision points that are used to prioritise and decide which mechanisms are to be applied
    - Dependencies
      - When one mechanism relies on another mechanism for its operation
    - Constraints
      - Restrictions one mechanism places on another

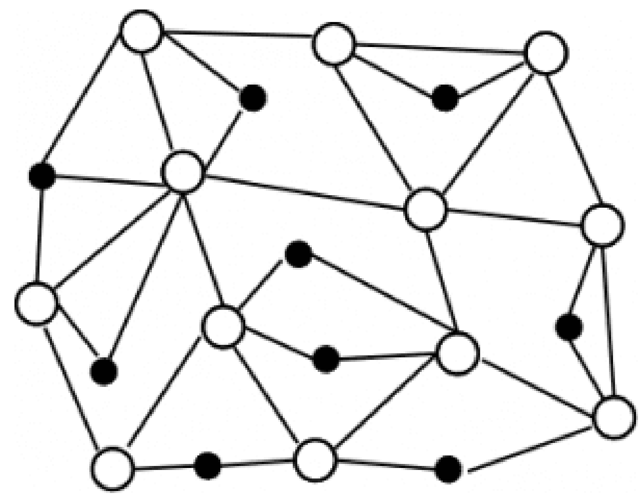
An example of an internal trade-off

Centralized network management is easier to implement and administer than Distributed network management, but is more prone to failure.



Centralized

Vs.



Distributed

An example of an internal trade-off

Centralized Network Management	
Advantages	**Disadvantages**
A single central server is quick and easy to deploy because you only have to manage one configuration without load balancing or orchestration.	A single master server presents a single point of failure on the network. If the master server goes down because of a bug or attack, or if you need to restart the server for maintenance, your entire network goes down.
You can easily add and remove client systems, users, and other servers without waiting for replication among decentralized or distributed servers.	All your valuable and sensitive data is stored and accessible from one server in a centralized network, presenting a security risk. If hackers get into your single DC, they can access everything from that one location, rather than needing to jump to different systems and servers to find everything they want.
Centralized networks are relatively inexpensive because you're using a limited number of servers, which means purchasing less equipment and fewer licenses.	Centralized networks are challenging to scale because there's a limit to how much computing power you can add to a single server. A central server can also create bottlenecks when your network traffic increases beyond the limitations of a single node.

An example of an internal trade-off

Distributed Network Management	
Advantages	**Disadvantages**
Distributed networks are extremely fault-tolerant because any server can fail independently without impacting the rest of the network at all—that server's functions are automatically shared among the other available servers.	Distributed network management is more expensive because it requires network orchestration tools to provide continuous load balancing and ensure that all nodes coordinate with each other for configuration and routing updates and changes to security policies.
Distributed networks are highly scalable because you can add new servers wherever they're needed at any time.	Distributed networks are more complicated to architect and implement, and not as many network engineers and sysadmins have hands-on experience working with them.
Distributed networks experience lower latency than other architectures because network processing power is evenly distributed among many nodes.	
On a distributed network, no single server controls all your enterprise's sensitive data and critical services. If a hacker breaches one node, they'll only be able to see the process and resources controlled by that individual server.	
A hacker can only inflict minimal damage to a server on a distributed network before your network orchestration solution redistributes network processes to a different server.	

An example of internal dependencies

	QoS	SLAs	Policies
QoS		QoS at network devices may need to enforce SLAs	QoS at network devices may need to enforce policies
SLA	Can SLA be enforceable via available QoS mechanisms		Do SLAs map to network policies?
Policies	Can Polices be enforceable via available QoS mechanisms	Are policies dependent on SLAs?	

QoS: Quality of Service
SLA: Service Level Agreement

An example of an internal constraint

- An easy to administer routing protocol such as RIP is not able to cope with large hierarchical network topologies

RIP: Routing Information Protocol

An example of an internal constraint

- **Routing Information Protocol (RIP)** is a dynamic routing protocol that uses **hop count** as a routing metric to find the best path between the source and the destination network. It is a distance-vector routing protocol that has an AD value of 120 and works on the Network layer of the OSI model. RIP uses port number 520.

- **Hop count** is the number of routers occurring in between the source and destination network. The path with the lowest hop count is considered as the best route to reach a network and therefore placed in the routing table. RIP prevents routing loops by limiting the number of hops allowed in a path from source and destination. The maximum hop count allowed for RIP is 15 and a hop count of 16 is considered as network unreachable.

Optimising Component Architectures

- We develop component architectures to support high priority flows.
- There needs to be a transparent relationship:

requirements → flows → component architecture

- In order to please your marker, you should make sure that you emphasise the link between these parts in your major report.

Reference Architectures I

- A description of the complete network architecture contains:
  - All component architectures being considered
  - Compilation of all internal and external relationships

Reference Architectures II

- Reference architectures combine these components and define external relationships between components (Remember that internal relationships are covered under each individual component architecture)

Reference Architectures III

- External Relationships
  - – Effects that architectures have on each other
- The balance of the reference architecture will depend on:

  ① Priorities given to functions in analysis stage

  ② Priorities given to flows

Reference Architectures IV

- Example
  - The requirements analysis determines that low delay and low jitter are the primary goals
  - Delay performance is affected by
    1. Network management
    2. Security
    3. Routing
  - Hence the final architectural decisions for these will need to be made so that they support the primary goal

Optimising Reference Architectures

- Performance and Security
  - Inspection and control of network traffic and access
  - As security mechanisms increase
    - Performance decreases
  - Security can reduce end-end performance of an application

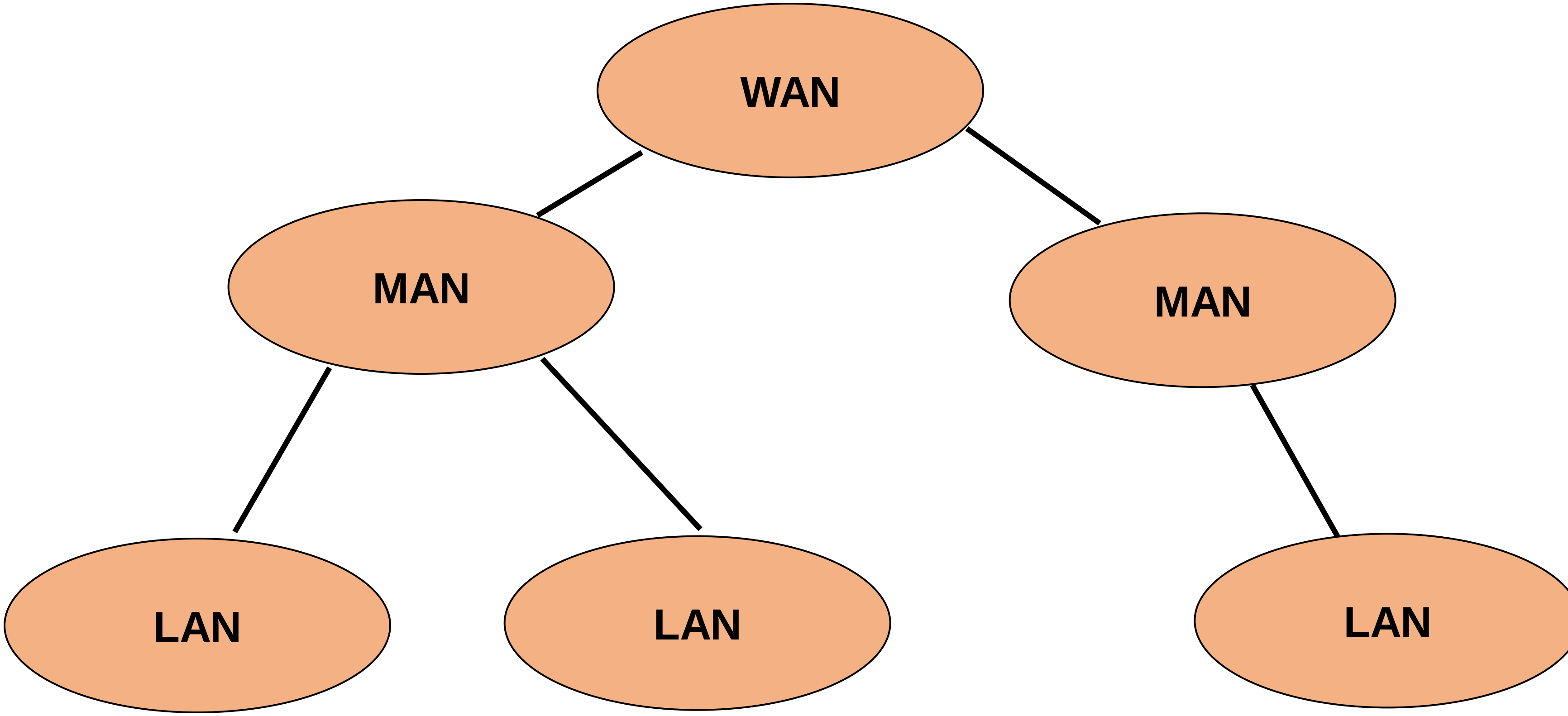
Optimising Reference Architectures

- Network Management and Security
  - Network management devices need access to network devices which represents a potential security “hole”.
  - Additional security required for in-band management will affect performance

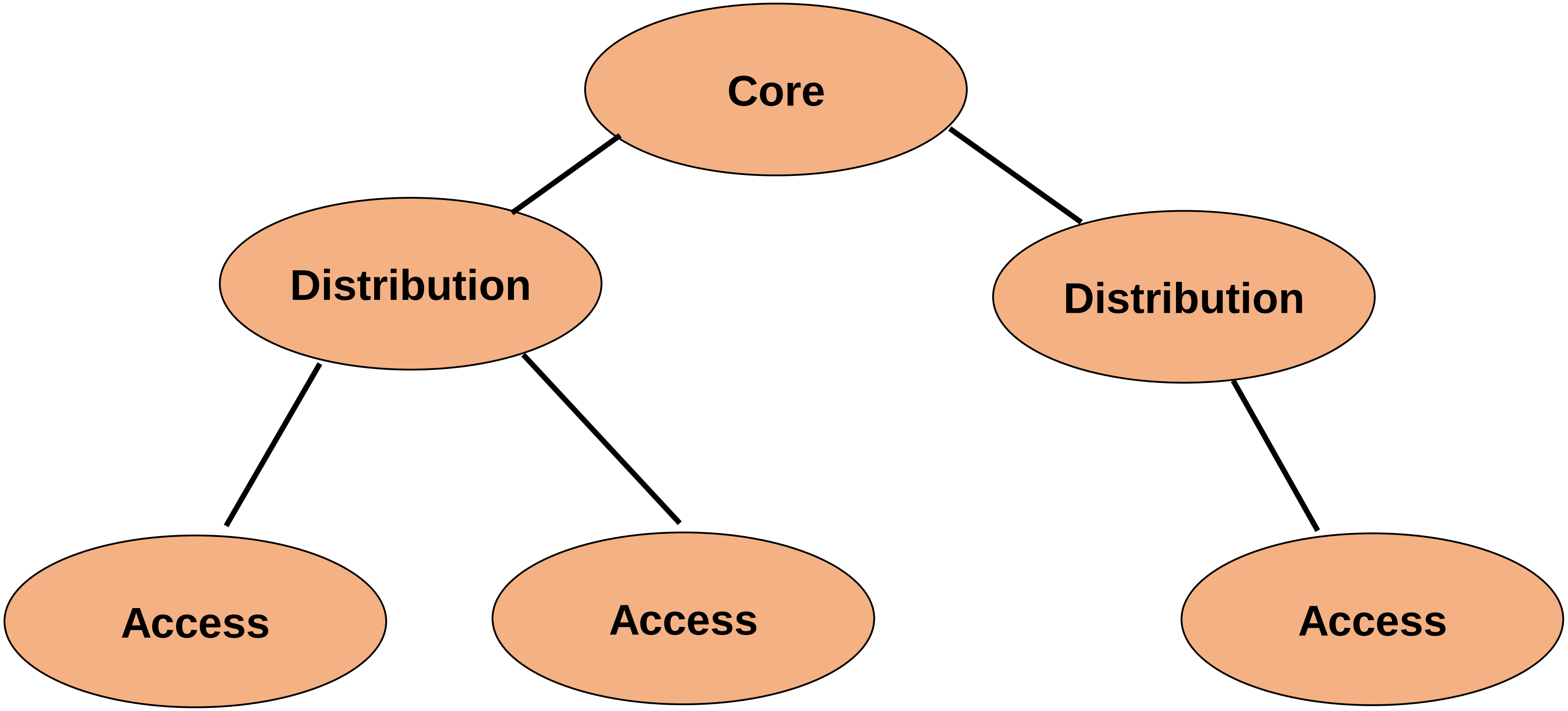
Architectural Models - topological

- McCabe suggests we begin with either of these two **topological** models.
  - LAN/MAN/WAN
    - Based on geographical size
  - Access/distribution/core
    - Based on function

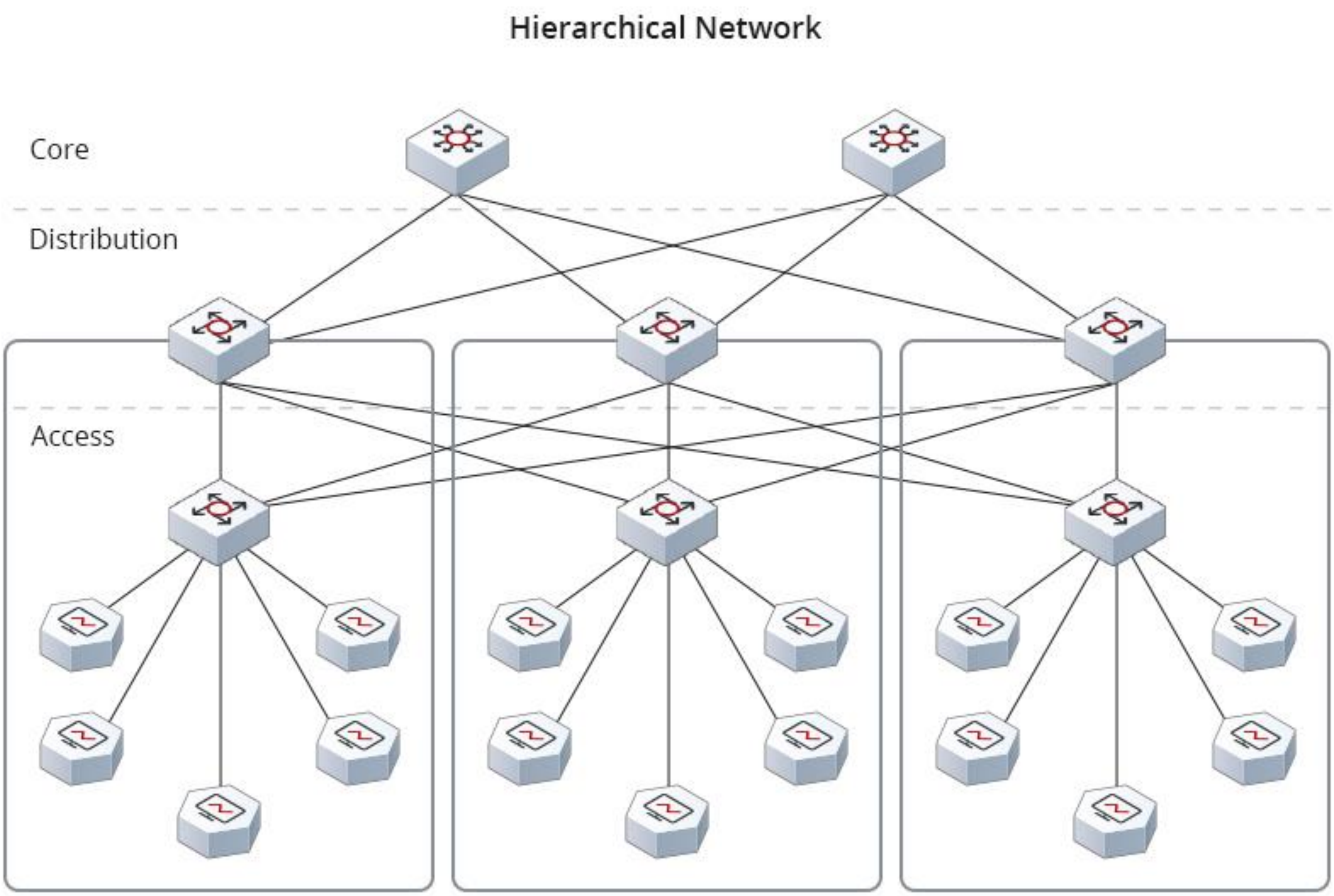
LAN/MAN/WAN



Access/Distribution/Core



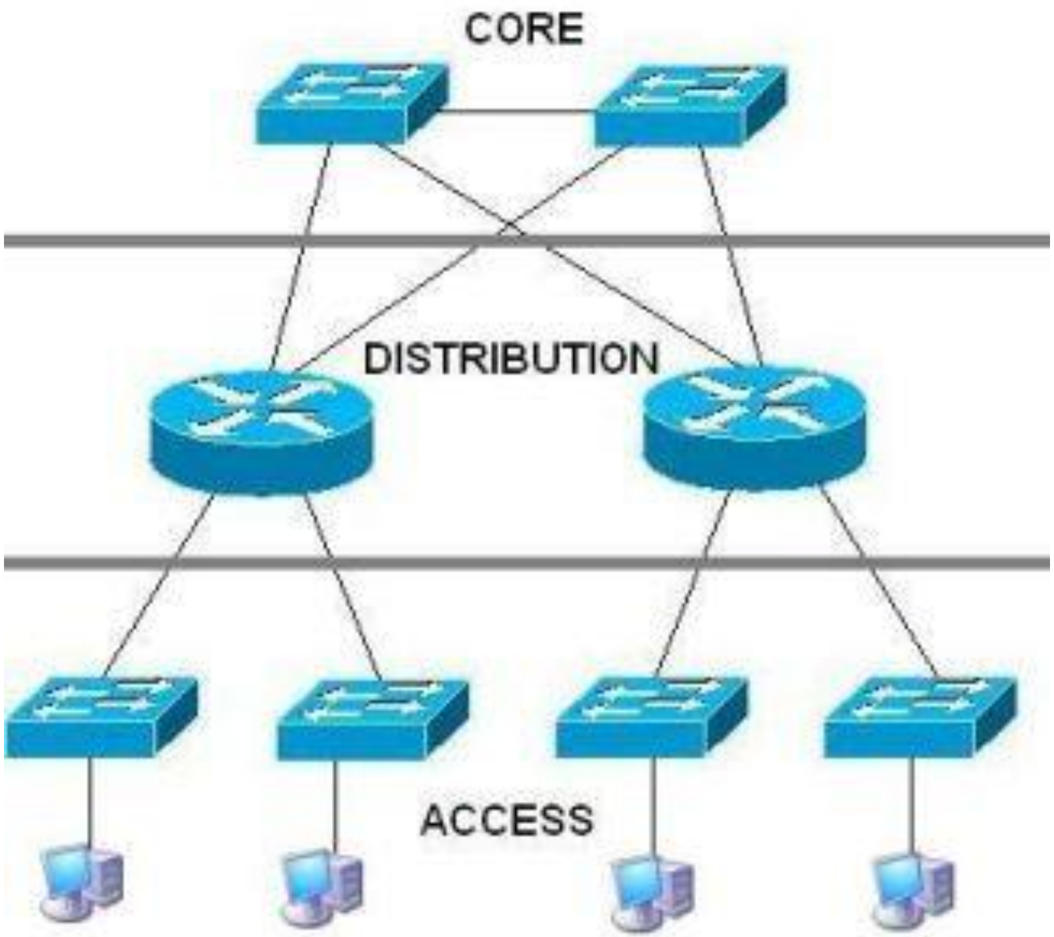
Access/Distribution/Core



Access/Distribution/Core

- Access (edge)
  - Where most traffic flows are generated and terminated
- Distribution
  - Where most traffic flows are aggregated and terminated for common services
- Core (backbone)
  - Provides transport for aggregates of traffic flows
- Demilitarised zones and External interfaces
  - Aggregation points for traffic flows external to the network

Hierarchy and Interconnection



Architectural Models – flow-based

- Flow-based models draw on requirements gathering work covered in Chapter 4

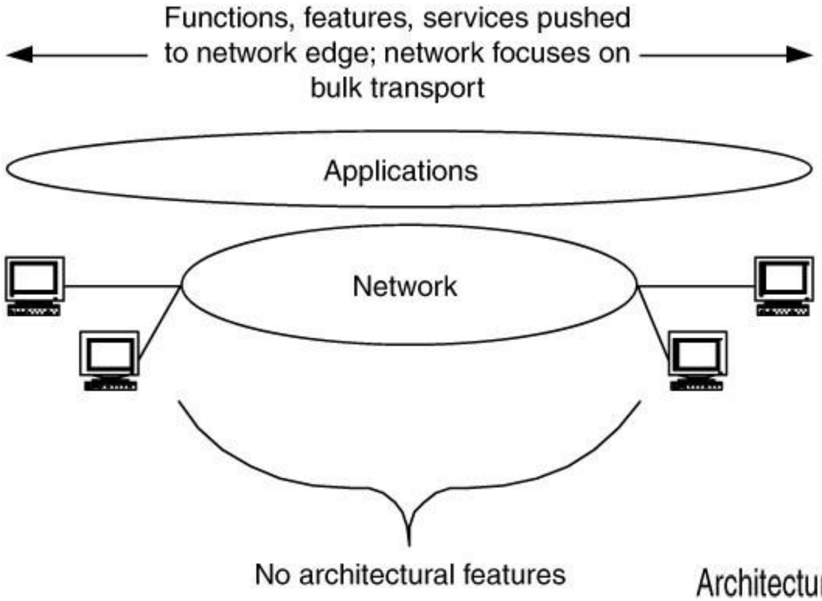
  ① Peer-to-peer

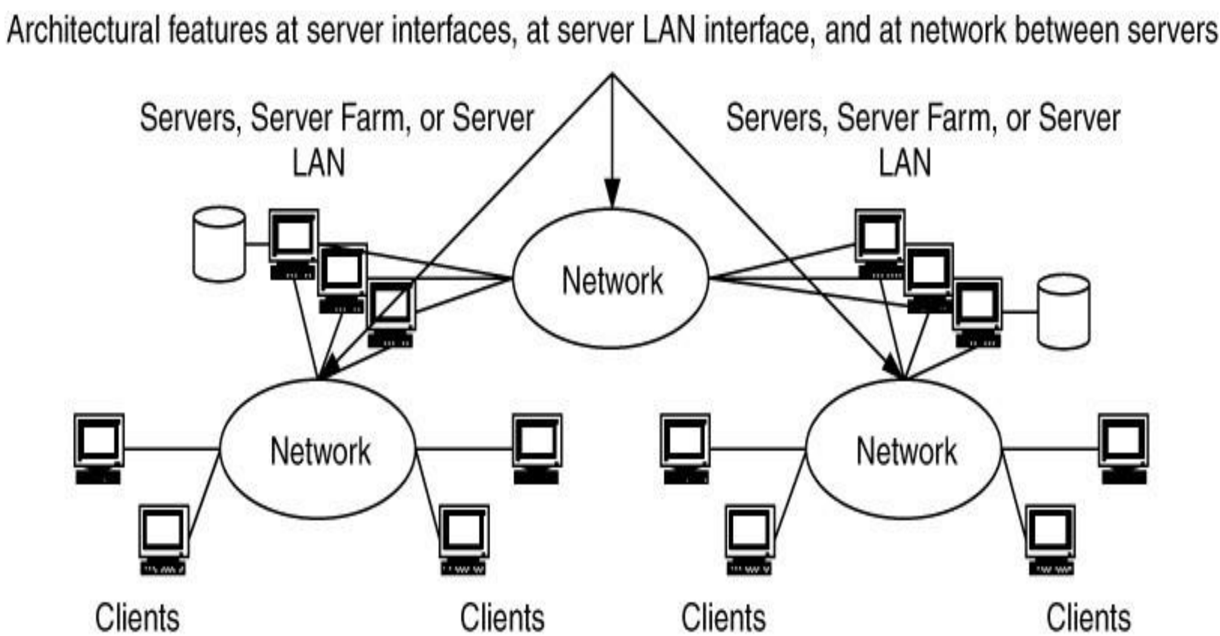
  ② Client-server

  ③ Hierarchical

  ④ Distributed computing

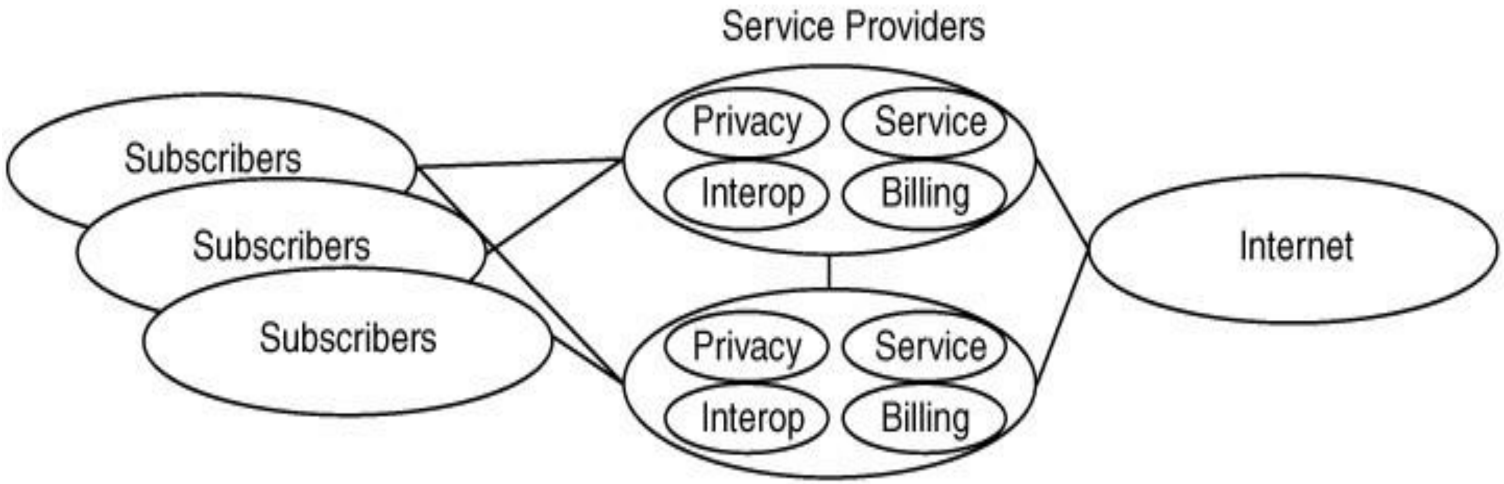
Architectural Models – flow-based





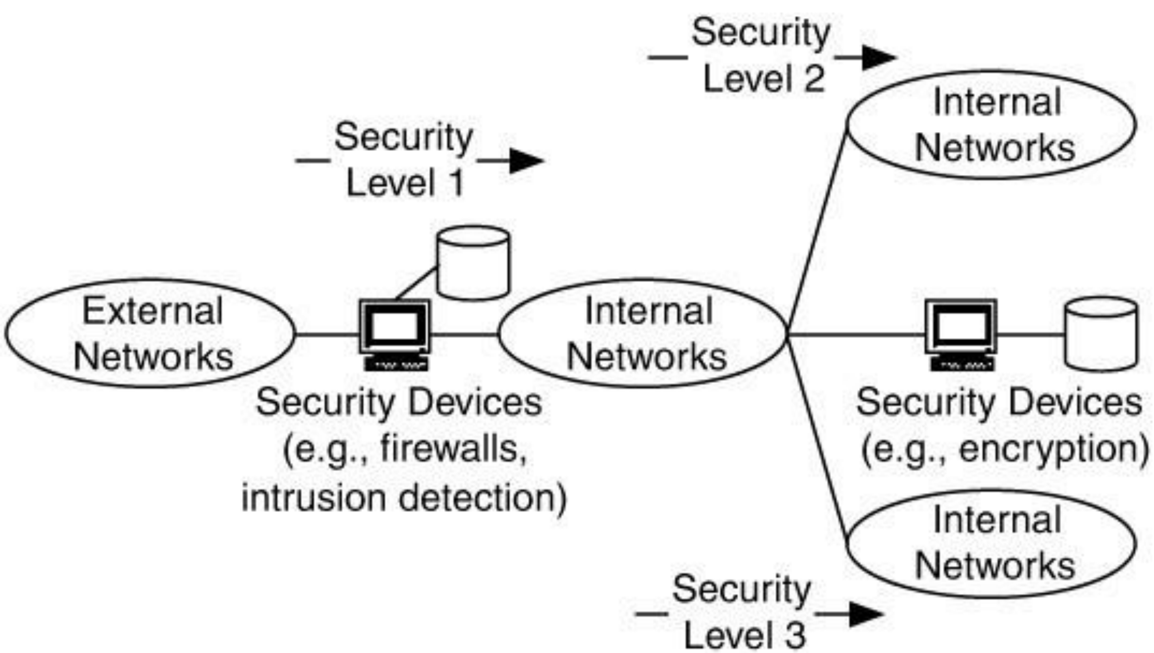
Architectural Models - functional

- Functional models
  - Service-provider architectural model is based on service provider functions
    - Privacy, security and service delivery



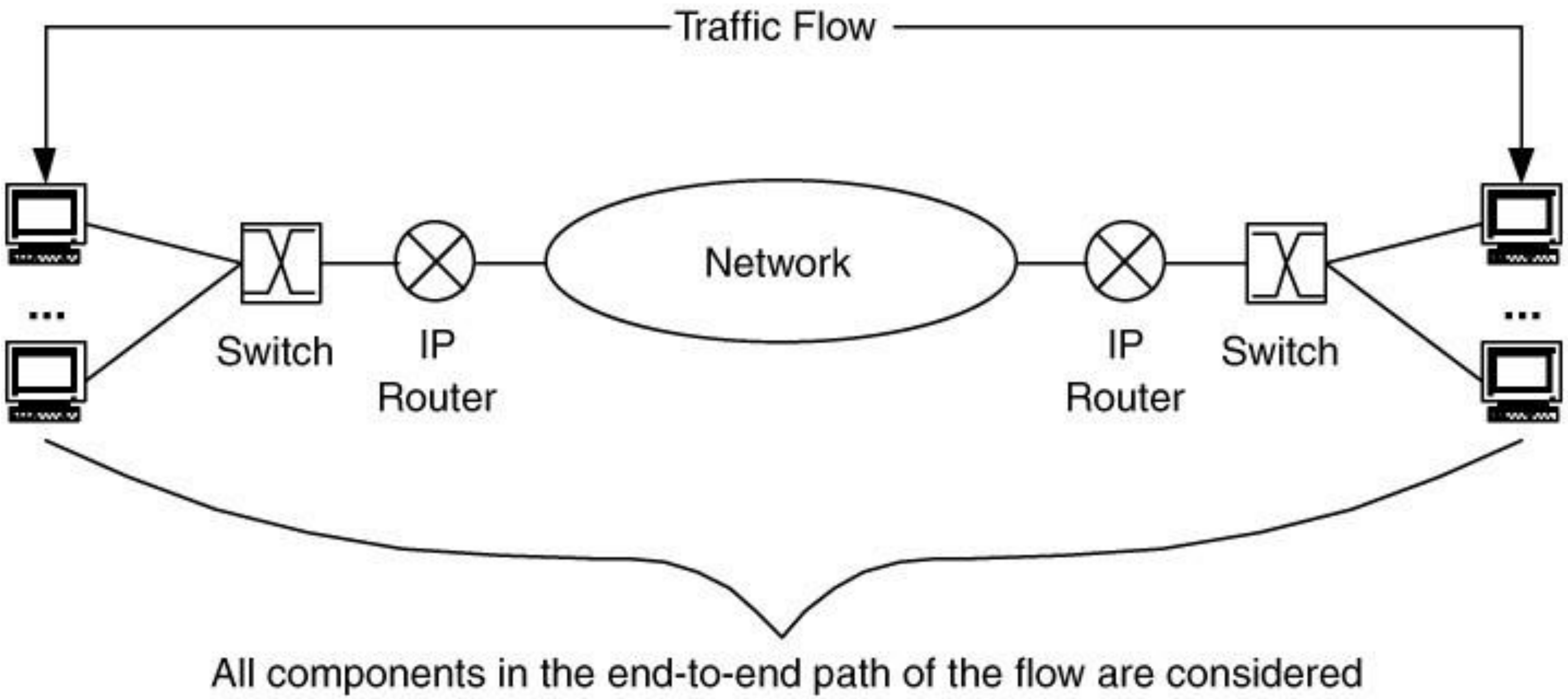
Architectural Models - functional

- Functional models
  - Intranet/extranet
    - Privacy, security, separation based on secure access



Architectural Models - functional

- Functional models
  - End-to-end – includes all the components in the path of the end-end traffic flow



Recipe for Using the models

- Start with a topological model
  - Add flow-based and functional models as required
- The result of this combination is the <u>reference architecture</u>

(Typically you will need a combination of the models discussed)