COSC2737
IT Infrastructure and Security

Load Balancing

Section 4.3

Overview

ITIS

- Load Balancing
  - What it is?
  - Why use it?
  - Where is it used?
  - Benefits thereof?

- Load Balancing Techniques
  - Techniques used

- Load Balancing Reference Architectures
  - Sun / Java Reference Architecture

Video Resources

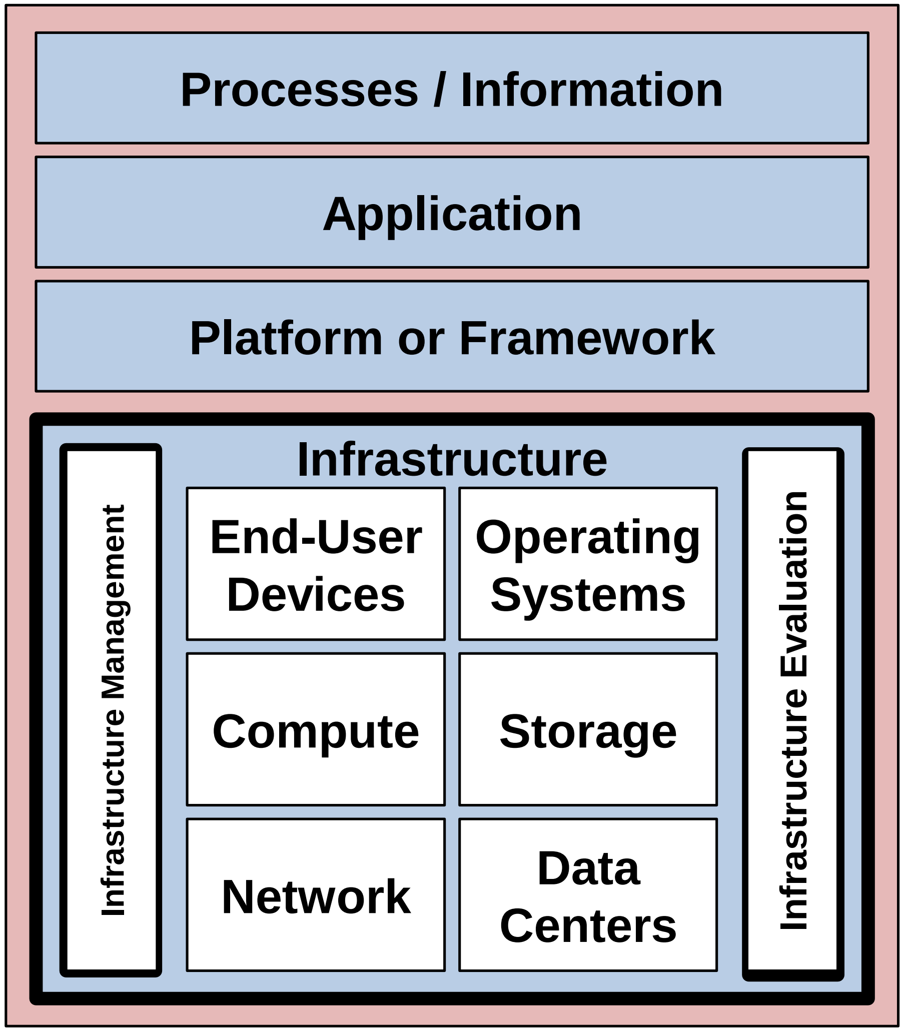
- **Apache Simple Load balancing with Apache**
  - https://www.youtube.com/watch?v=se4PhIwyWLw
- **How to Configure NGINX as a Load Balancer -**
  - **https://www.youtube.com/watch?v=v81CzSeiQjo**
- **Advanced**
  HTTP Load Balancing and Sticky Sessions With Docker and Apache
  - https://www.youtube.com/watch?v=6gDW56dS8nU

Introduction

ITIS

- Load Balancing is an infrastructure optimisation that in the diverse cloud requires…
  - management
  - evaluation

… that is optimum for the mix and volume of business applications.



Load Balancing (LB)

- Load balancing (LB) is the process of distributing load across multiple devices or services in an equitable way

  The most common reason is

  - To establish uniform ***work*** across application servers (AS's)
  - To establish uniform ***resource*** use across the resource domain

- Load Balancing can apply to any kind of work,
  - Web Servers / Services
  - Terminal Services
  - CPU cycles,
  - Bus Arbitration / Throughput…

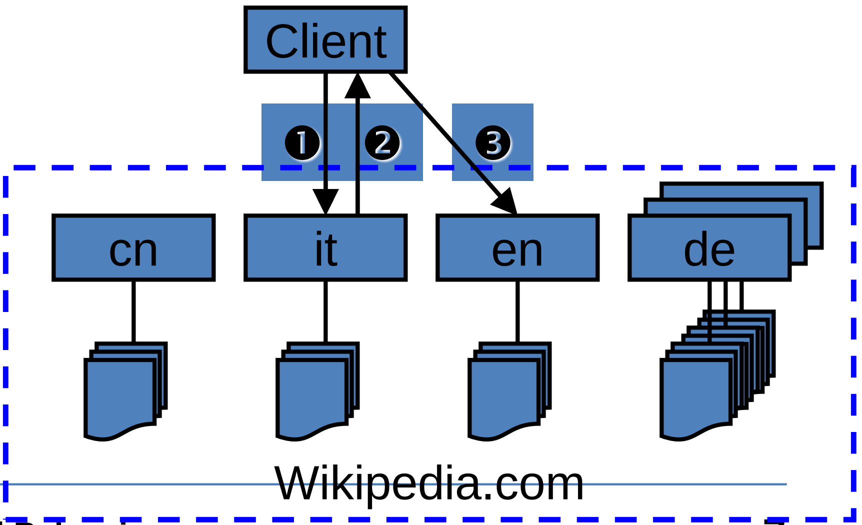
Why LB?

- Most enterprise class applications require scalable LB of some sort,
  - hardware and networks have finite resources; it is exhaustible.
  - when it runs out and more is needed,
    - How do we add it?
    - How do we tell the world about it?
    - Do we need to tell the world?
- Again, we see the need for caching.
- Recall the Caching Hierarchy.

Forms of Load Balancing

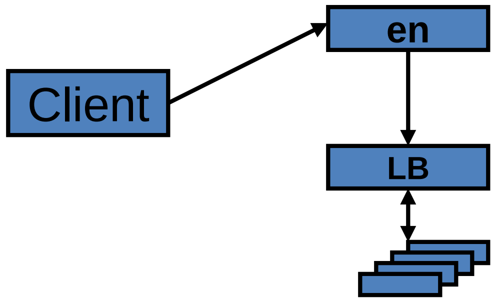
ITIS

- **Non-transparent**
  (client effectively selects server)
  - There may be occasions when a client selects the site (eg http://en.wikipedia.com/ may be a different site to http://it.wikipedia.com/ which might be located in Italy)
    - But each can have links to the other(s)
  - Unlike content-negotiation where multiple DNS entries point to the same machine, and Apache selects the document tree internally, here DNS is used to select the machine via name resolution.
  - Distribute the load:
    - Geographically
    - Geometrically (as in Grid Computing)
    - Functionally (eg secure servers)
  - An Italian page might refer to an English page when the Italian does not exist.



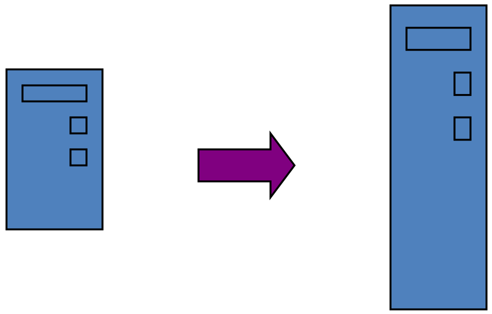
Forms of Load Balancing

- **Transparent**
  (client sees only one web site)
  - App Scaling
    - As more resources are needed to cope with increasing traffic, LB distributes load across multiple servers in order to scale applications.
      - Vertical scaling
      - Horizontal scaling
  - High availability
    - In mission-critical apps, a level of redundancy allows a site to remain up even if multiple components fail
  - Dynamic Reconfigurability
    - Allows a site to change its configuration frequently if necessary without telling the world about it
- Can be Both
  - The English Wikipedia could be non-transparent for language selection, but transparent within a language



Vertical Scaling – Scaling up

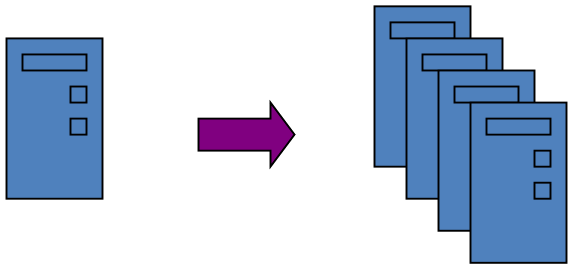
- Adding more hardware resources to the same machine often means more CPU/s and Memory
  - Example: 8 way vs 4 way machines (8 vs 4 CPUs)
- This is usually very expensive ($$)
- Easy to implement (generally no change required in your application)
  - But you still have a single point of failure



Horizontal Scaling – Scaling out

ITIS

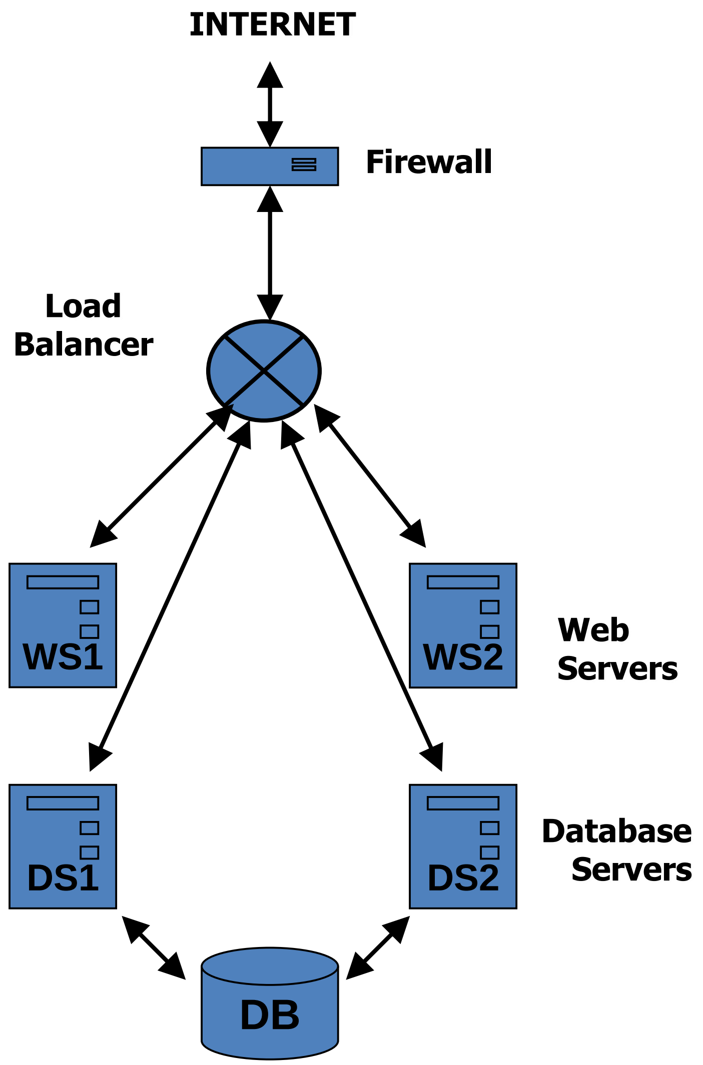
- – Adding more machines to the mix, generally cheap hardware
- – Cheaper – at least more linear expenditures
- – Harder to implement (much harder than vertical)
  - May add another protocol layer between machines
- – Many points of failure but can usually handle failures elegantly due to redundancies



A Load Balanced Application

- The diagram shows a simple load balanced application
- Balanced at every tier
  - Web Servers
  - Database Servers
    - The Web Tier uses the load balancer to talk to the DB Servers.

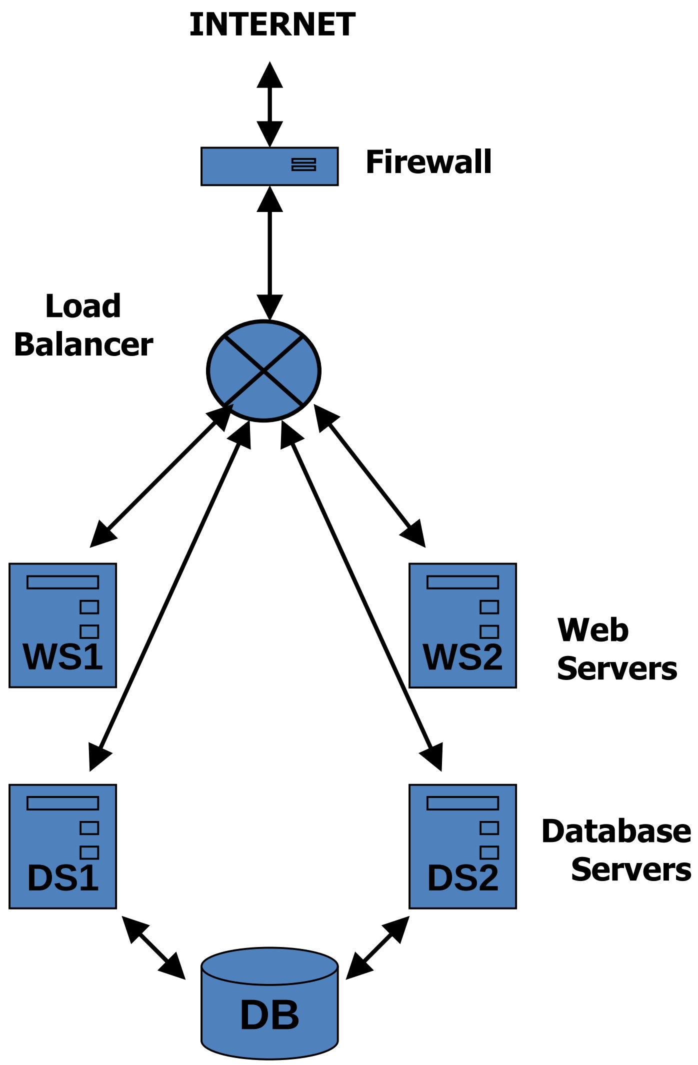
- Q: How many IP Addresses do you need for the whole application?



A Load Balanced Application

Q: How many IP Addresses do you need for the whole application?

- A: A minimum of 4
  - One for each WS
  - One for each DB
  - One more..?
    - For which component?



Introducing: Load Balancers

ITIS

- Given what you know about DNS Servers
  - How many IP addresses can you bind to one name?
  - So if you have a web application on multiple application servers… how does a client decide which component in the application to send to?
    - Does the client need to make that decision?
    - …if not so who makes it?

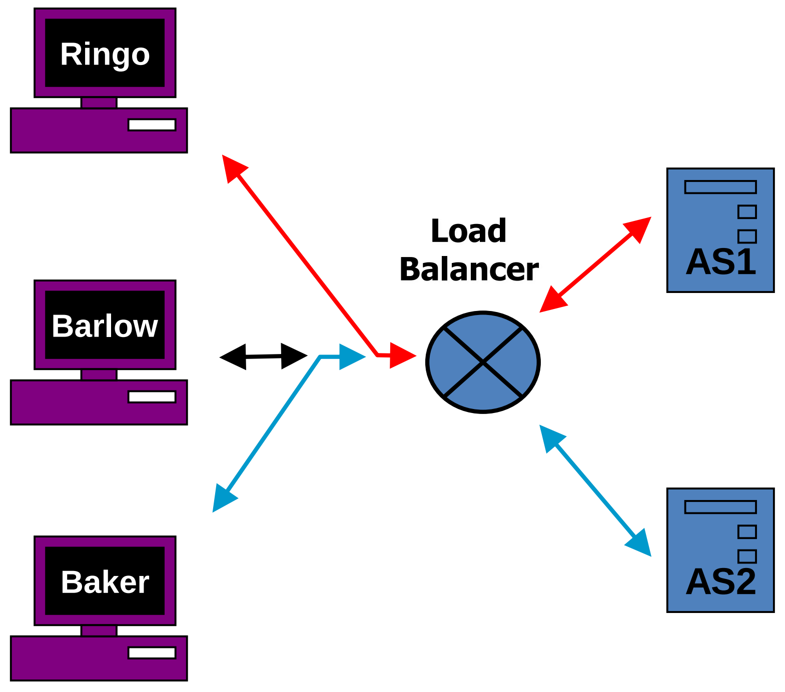
Introducing: Load Balancers

ITIS

- The DNS name/entry is usually bound to the Load Balancer IP Address (each load balancer needs an IP address).
  - Q: Why would *this* device require an IP address?

- The load balancer usually just acts as a forwarding device, and forwards packets (that come in on port 80; HTTP Requests) to/from a client to/from an Application Server
  - So it is the *load balancer*, not the client, that decides which part of the application a packet goes to.
  - …so the client needs to talk to the load balancer, hence the LB needs an IP address and a DNS entry.

Load Balancer Example



- Our load balancer has a single entry point from the internet, and two application servers
- Request Sequence: Ringo, Baker, Barlow
- So Ringo's request is forwarded to Application Server AS1, and Baker's request is forwarded to AS2
- Then Barlow's request arrives.
  - Where does it go?

Load Balancer Example

- In the previous schematic we see that each client is forwarded to a different application server.
  - The requests are thus balanced by the Load balancer (follow the colored lines)
  - So how do we decide which AS to use?

- Algorithms of Load Balancing
  - The simplest algorithm is called Round-Robin (everyone gets a turn).
    - In this case Barlow would have its request forwarded to the next available Application Server (AS)?

Load Balancer Algorithms

ITIS

- Load Balancers decide how to balance a load based on different algorithms.
  - These are essentially rules that govern the overall behavior of the Load Balancer.
  - These rules can be applied to
    - **External events**
      - Network traffic, traffic distribution, etc
    - **Internal events**
      - AS response times, AS status (down,up,…)

For more detail, see:

"Understanding CSM Load Balancing Algorithms"
http://www.cisco.com/en/US/products/hw/modules/ps2706/products_tech_note09186a00801adbde.shtml

LB Parameters

Some of the most important parameters affecting LB are:

- *Node availability*
  - A list of nodes that the LB can forward requests to,
  - The node status (busy, free, offline, resetting, …).
    Can be as simple as the LB pinging the AS.

- *Advance knowledge of traffic behaviour*
  - The LB knows a traffic pattern in advance
    (eg central ATM services, some web services)
  - Can make resources available in advance or schedule subsequent transactions (eg balance update audit trail)

- *Statistical estimate of traffic behaviour*
  - Using network traffic modeling algorithms
  - Trying to predict future needs based on present trends

LB Parameters (2)

ITIS

Some other parameters affecting LB are:

- ***Propagation Delay (PD)***
  - The act of generating statistics (ie network stats or load stats) on a server takes time,
    - thus this adds latency to requests.

- ***Session Persistence ('Stickiness')***
  - Most web pages generate multiple TCP sessions, which should all normally go to the same server in order to enable memory caching.
    - Q: Why is this required? Is this ALWAYS required?
  - May affect efficiency of LB algorithm if stickiness not properly distributed
    - 'failover' to a different server if a server goes down is not trivial

- ***Transaction complexity***
  - Operations may range from a simple static HTML read to a multi-record DB update.
  - Multiple AS updates to databases may affect LB algorithm non-trivially

LB Algo: Random (R)

ITIS

- The simplest load balancing algorithm is Random
  - Directs the network connection to any available server,
  - Treats all *servers* as equal, regardless of the number of connections or response time.
  - Treats all *network transactions* as equally complex
  - Has no memory, so it treats all transactions as idempotent (same effect no matter which server, or sequence)
    - True for HTTP, and HTTP GETs.
    - Q: Is it true for HTTP forms? True for session cookies?
- Makes the grandiose assumption that all traffic is equal which is clearly false, so not appropriate for web use.

LB Algo: Round Robin (RR)

ITIS

- The next simplest load balancing algorithm is **Round-Robin**
  - Directs the network connection to the next available server, allocated in some sequence
  - Treats all servers as equal, regardless of the number of connections or response time.
  - Treats all network transactions as equally complex
  - Has no memory, so it treats all transactions as idempotent (same effect no matter which server, or sequence)
    - True for HTTP, and HHTP GETs.
    - Q: Is it true for HTTP forms? True for session cookies?
- It is simple and sometimes very effective because no propagation delay or caching hinders the algorithm.

LB Algo: Least Connections (LC)

- The ***Least Connections*** method directs network connections to the server with the fewest connections.
  - Although it may not be intuitively obvious that the mathematical predictor would provide effective load balancing, in fact, it is quite successful, according to Cisco.
  - On Web sites where there is a collection of servers with similar performance, the predictor is effective in smoothing distribution when a server becomes bogged down.
  - On Web sites where there are large differences in the capacity of various servers, the predictor is also very effective.
    - In maintaining the same number of connections to all servers, those servers that are capable of processing (and thus terminating) connections the fastest receive more connections over time. A server deemed to be twice as powerful as another server receives about twice as many connections per second.

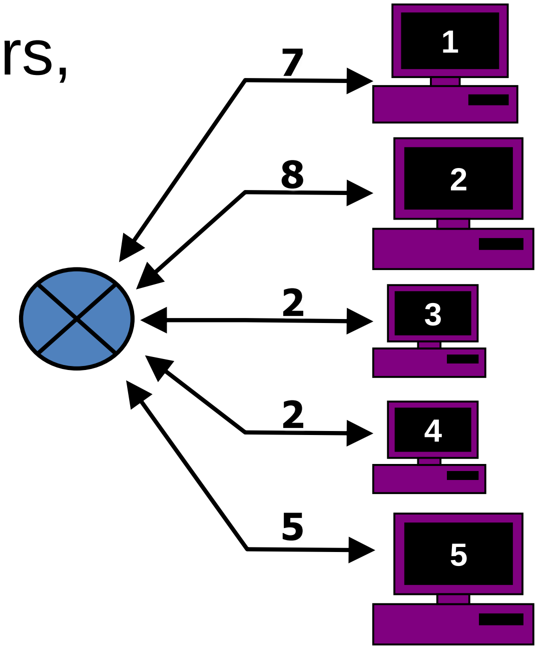
LB Algos: Weighted RR / LC

- The ***weighted*** method allows you to assign a performance weight to each server.
  - Administrators can assign a weight to each real server, and the Load Balancer uses this weight to determine the percentage of the current number of connections to give each server.
  - Servers with a higher weight value receive a larger percentage of connections at any one time.
  - Weight usually allocated on server properties such as
    - Capacity, CPUs, etc…

LB Algos: Weighted Methods

ITIS

- EG: In a configuration with 5 servers, the percentage of connections is calculated as follows:
  - Server Number of Connections
    - Weight of server1 = 7
    - Weight of server2 = 8
    - Weight of server3 = 2
    - Weight of server4 = 2
    - Weight of server5 = 5 .
  - Total weight of all servers 24



- This distribution results in server 1 getting 7/24 of the current number of connections, server 2 getting 8/24, server 3 getting 2/24, and so on.
  - If a new server, server 6, is added with a weight of 10, it receives 10/34, and so on.

LB Algorithm: Pros and Cons

ITIS

- **Random**
  - *Pros:* Extremely simple to implement
  - *Cons:* Too simple to actually work well ☺
- **Round Robin**
  - *Pros*: Better than random allocation because the requests are equally divided among the available servers in an orderly fashion.
  - *Cons*: Round robin algorithm is not enough for load balancing based on processing overhead required and if the server specifications are not identical to each other in the server group.

LB Algorithm: Pros and Cons (2)

ITIS

- **Weighted Methods**:
  - *Pros*:
    - Adapts according to the capacity of the servers in the group.
    - Useful in app-specific cases involving function-specific servers in a cluster
  - :
    - Does not consider advanced load balancing requirements such as processing times for each individual request.

LB Algorithm Adaptation

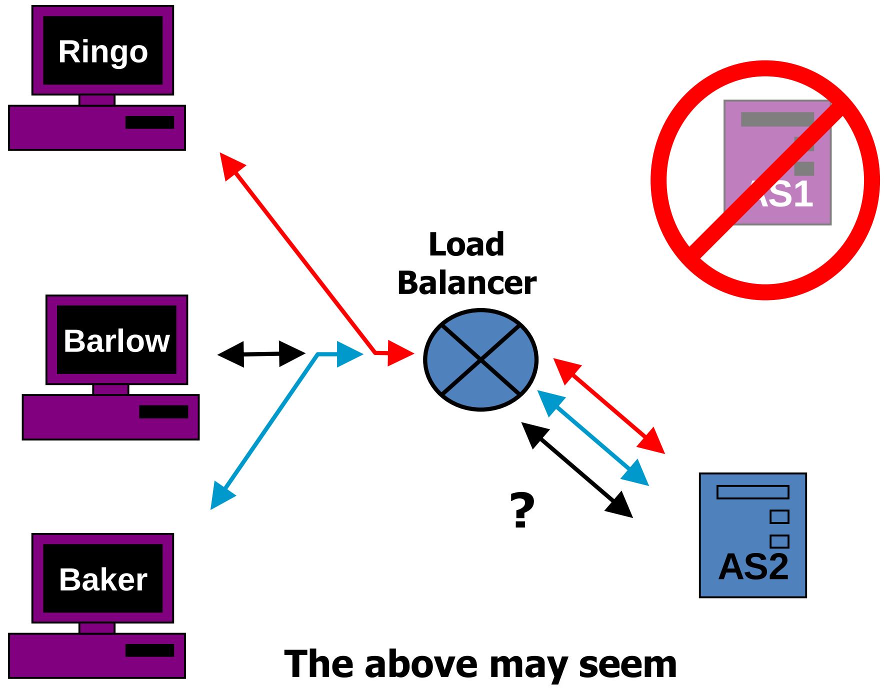
ITIS

- An essential component of load balancing is LB adaptation (also called intelligent forwarding)
  - For example, the simple Round-Robin method on its own as described earlier is not adaptive.
    - It treats all network transactions as independent
    - It treats all network transactions identical in complexity
    - It treats all application servers as and identical
    - It does not distinguish between transactions requiring database server access, from those that do not.
  - These are significant weakness of this algorithm.

The need for intelligent forwarding

ITIS

- What happens to the client requests if some of the application servers are down?
- What happens if the network destination is unreachable, or the network is saturated, or worse yet the network on half the application servers is congested and the other half of the application servers is relatively load free?
- Will the client experience be uniform for all clients (Ringo, Barlow, Baker) ?



The above may seem obvious, but what if there were 10 AS's or which 3 were down?

The Health-Check

- Every load balancer has a list of nodes (eg. available web servers) it can forward requests to. This list is usually availability based
- Availability may be as simple as the load balancer pinging each Web Server or requesting an Static HTML document from the Web Server
- So the load balancer can keep this list up to date and adapt the load dynamically.

Load Balancer Issues: Sessions

- An important class of web transactions are when these involve sessions.

- When a transaction is session-based, the session information must be kept somewhere.
  - If it is kept on the AS, what happens when the next transaction goes to a different AS?
  - If it is kept on the LB and transferred to each AS as needed, what happens when the LB goes down – what failover is available?
  - If it is kept on the client, then it persists as long as the client persists, which is the desired behaviour anyway.

- Q: How can we store session information on the client in a HTTP transaction?

Load Balancer Issues: Sessions

ITIS

- Q: How can we store session information on the client in a HTTP transaction?
- Q: What if we don't, then what do we need to ensure?

- A load balancer required to maintain persistence must support sticky sessions.
- A sticky session specifies that once a session is created by a given server, subsequent requests from the user will continue to be routed to that same server in order to preserve session information.

  – The question comes back to idempotence:
  – In a session, all transactions are not equal since they depend on session information (eg trans time, seq, result, etc…).

- We can make them all equal by storing the session information with the transaction. This may not always be possible.

- Q: How did we do that before, in this course?

Load Balancer Issues: Sessions

ITIS

Q: How can we store session information on the client in a HTTP transaction?

- A: Cookies!!! ☺

- Two scenarios are possible:
  - A load balancer may not need to support sticky sessions if all the session information is in the transaction itself and this is carried through to all resources that need it (eg database server)
    - Idempotent → NO sticky sessions
    - LB algorithm free to allocate server without regard to session
  - A load balancer can be deployed *with* cookie based persistence in which case, the cookie is used to relay session information, that the load balancer uses to redirect to the server that created the session.
    - Not Idempotent → Sticky Sessions
    - LB algorithm constrained in its resource allocation

LB: Putting it all together

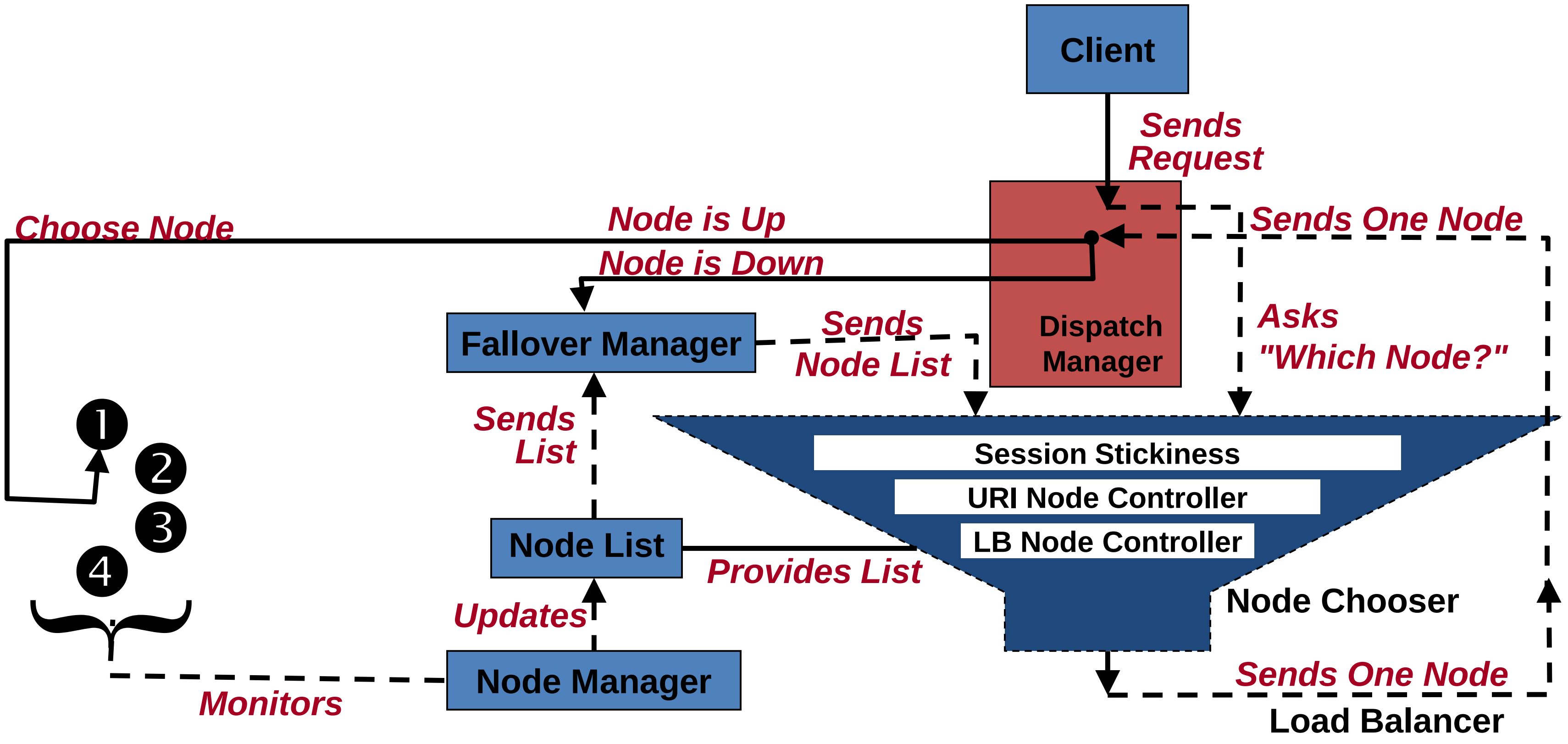
- For session-enabled LB (LB assigns cookie)
  1. ***REQ:*** **Client → LB**
  2. ***REQ:*** **LB → AS (next available) → say, AS3**
  3. ***RESP:*** **AS3 + (session saying AS3) → LB_3**
  4. ***RESP:*** **LB_3 → $Client_3$**

- In a subsequent transaction
  1. ***REQ:*** **$Client_3$ → LB (sees session sayung AS3)**
  2. ***REQ:*** **LB → AS3**
  3. ***RESP:*** **AS3 + (session saying AS3) → LB_3**
  4. ***RESP:*** **LB_3 → $Client_3$**

LB: Sticky Sessions

ITIS

- Here is the entire concept of how Load Balancers work and how servers are clustered with load balancers in order to provide high availability.



LB: Putting it all together (2)

- Q: What happens if the Web Server the client is 'stickied to' becomes unavailable half way through a client request
- A: In other words, what happens if session info is lost?
  - Either
    - the client is forwarded to a new AS if session can be transferred,
    - the client is logged out and a new session is started.
  - This is up to the implementation of the Load Balancer and/or application.

- Some applications keep session information in a database thus allowing another web server to restart the session if needed.
  - This may add to the load non-trivially

Summary

ITIS

- In summary, Load Balancing:
  - can be transparent or non-transparent
  - is used to ensure HA/CA
  - is measured using MTBF and MTTR
  - is implemented using one of 5 algorithms
  - is adaptive in order to work in practice
  - uses sessions internally, or as client cookies
  - is often described using "the nines"