Load Balancing & Redundancy

Introduction

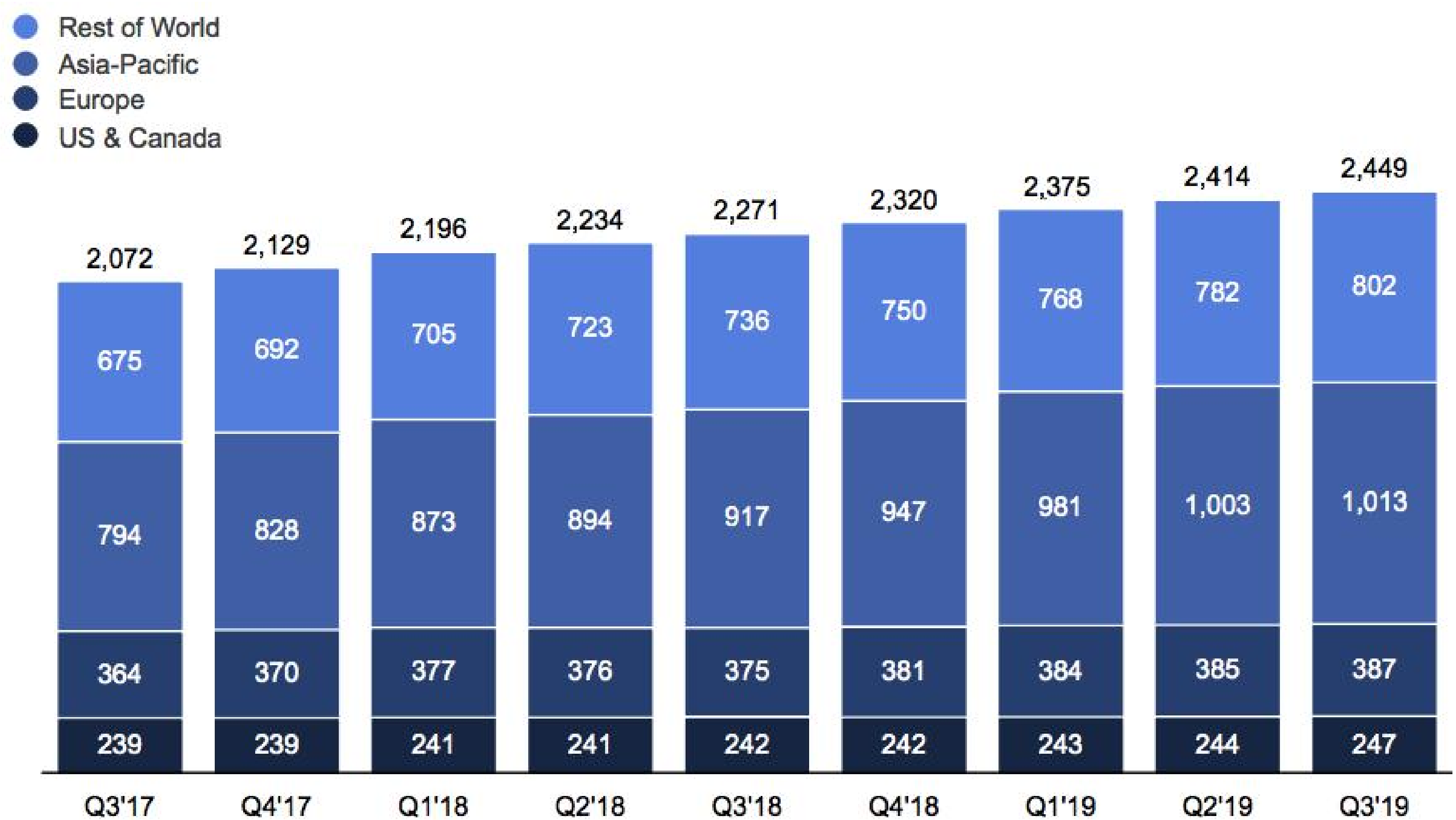
- Explosive Growth of the Internet

World Regions	Population (2020 Est.)	Population % of World	Internet Users 31 Dec 2019	Penetration Rate (% Pop.)	Growth 2000-2020	Internet World %
Africa	1,340,598,447	17.2 %	526,374,930	39.3 %	11,559 %	11.5 %
Asia	4,294,516,659	55.1 %	2,300,469,859	53.6 %	1,913 %	50.3 %
Europe	834,995,197	10.7 %	727,814,272	87.2 %	592 %	15.9 %
Latin America / Caribbean	658,345,826	8.5 %	453,702,292	68.9 %	2,411 %	10.0 %
Middle East	260,991,690	3.9 %	180,498,292	69.2 %	5,395 %	3.9 %
North America	368,869,647	4.7 %	348,908,868	94.6 %	222 %	7.6 %
Oceania / Australia	42,690,838	0.5 %	28,775,373	67.4 %	277 %	0.6 %
WORLD TOTAL	7,796,615,710	100.0 %	4,574,150,134	58.7 %	1,167 %	100.0 %

http://www.internetworldstats.com/stats.htm

Sites receiving unprecedented workload

- FB has ~33% of world pop (FB Q3 2019)



Sites receiving unprecedented workload

– Twitter: ~6000 Tweets/second (340M users)
– Youtube: 5B views/day, Up: 300h/min (2B users)
– Instagram: 40B photos, 80M/day (1B users)
– Facebook: 300M photos day

Sites receiving unprecedented workload

- Even small web services require to consider scalability and availability concepts

- **Scalability:** In order to keep with increasing demand

- **Availability:** In order to maximize service operation

Definitions

Scalability

- The ability of a software to operate over an increasing set of resources

- **Horizontal**: higher number of resources
  - software may need to support distributed operation
- **Vertical**: more powerful resources
  - generally transparent to software

Definitions

Availability

- The percentage of time a software is providing a service do consumers (users, other software)

(To be detailed later in the course)

Definitions

load balancing

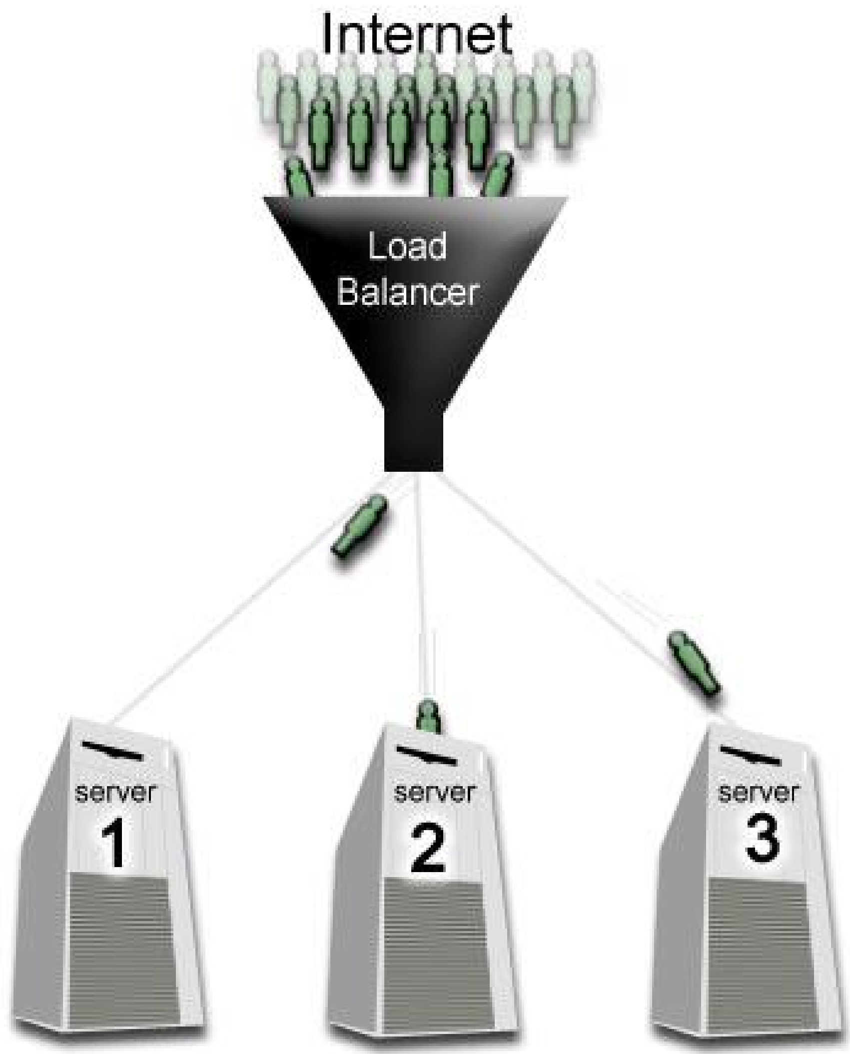
- A technique to spread work between many computers, processes, disks or other resources in order to get optimal resource utilization and decrease computing time.

Definitions

Load balancer

- Virtual Server (also referred to as vserver or VIP) which, in turn, consists of an IP address and port.
  - A load balancer can be used to increase the capacity of a **server farm** beyond that of a single server.
  - It can also allow the service **to continue** even in the face of server down time due to server failure or server maintenance.
  - virtual server is bound to a number of physical services running on the physical servers in a server farm.

Load Balancer



Virtual Server

- **Different virtual servers** can be configured for different sets of physical services, such as TCP and UDP services in general.

- Application specific virtual server may exist to support HTTP, FTP, SSL, DNS, **provided** by an additional server.

- Load balancing methods **manage the selection of an appropriate physical server** in a server farm.

Load Balancers

- Load balancer maintains the system in a state where load on a node is below it's target
  - **Load**: could be storage, bandwidth, etc.
  - **Target**: the load a node is willing to take (ex. capacity, avg. util. + slack)

- Assumptions
  - Nodes are cooperative
  - Only one bottlenecked resource

Why to Load-balance?

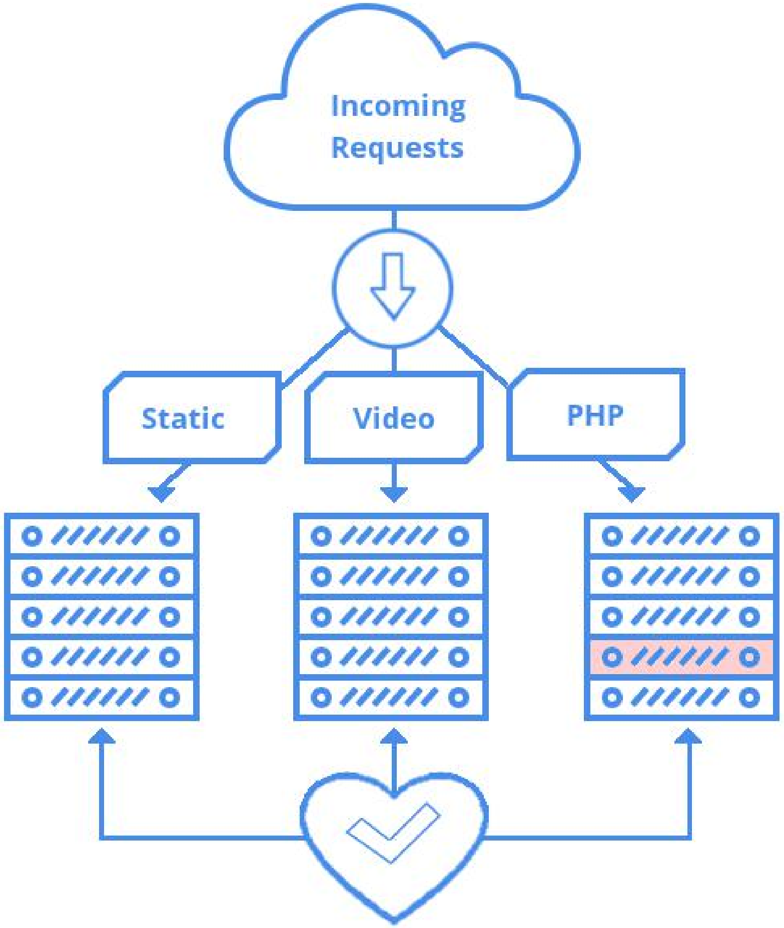
- Time to serve request is bound to capacity of a single CPU + storage latency + database latency
  - Dynamic Content: **Tens** of reqs/cpu/second
  - Static Content: **Thousands** of reqs/cpu/second

- Options:
  - Buy faster CPU (Vertical scalability)
  - Buy more CPUs and **Load Balance** (Horizontal scalability)

Why to Load-balance?

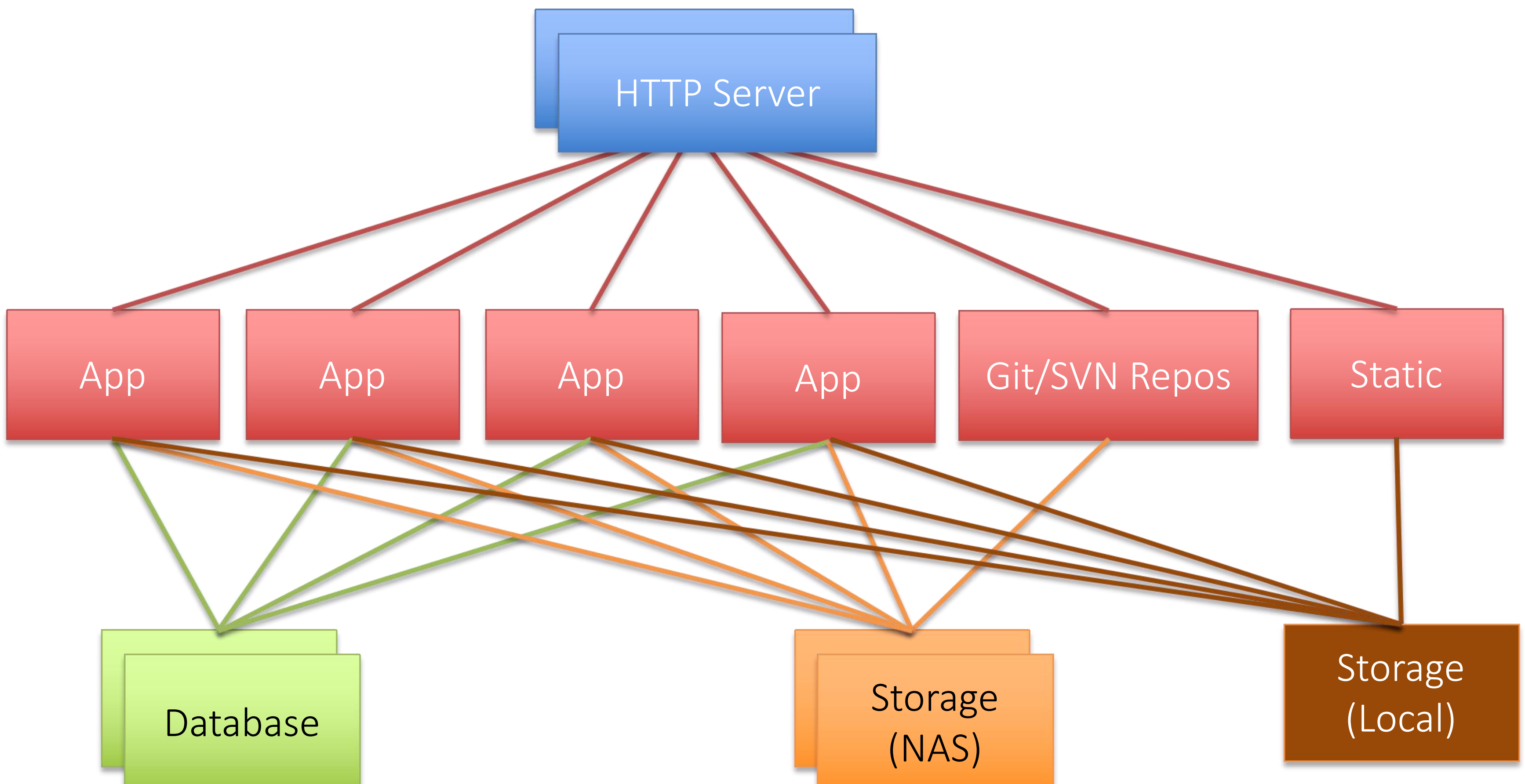
- Scale applications / services
- Ease of administration / maintenance
  - Easily and transparently remove physical servers from rotation in order to perform any type of maintenance on that server.
- Resource sharing
  - Can run multiple instances of an application / service on a server; could be running on a different port for each instance; can load-balance to different port based on data analyzed.

Content Based Load Balancing

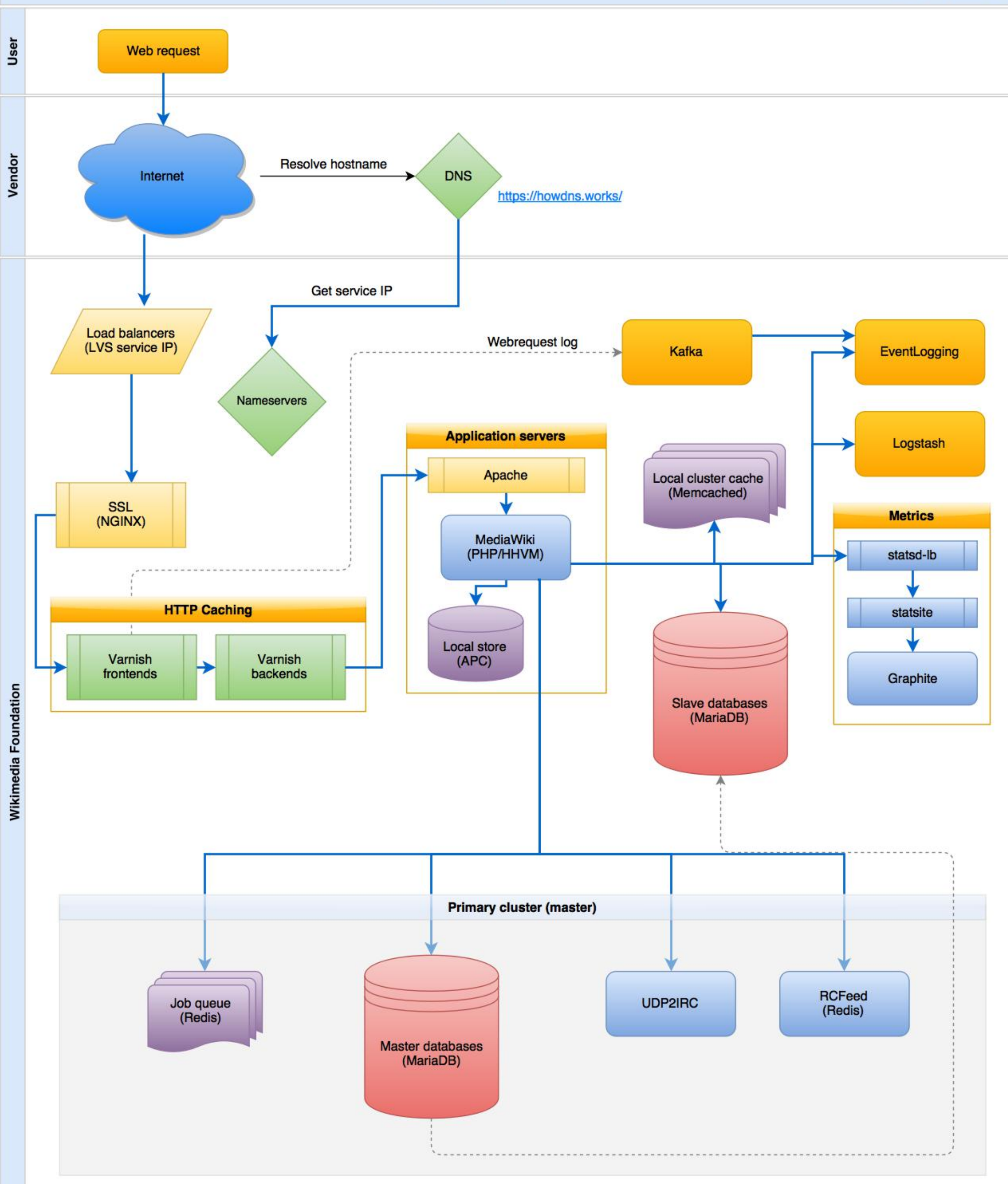


Applied even to small services

CodeUA



Wikipedia


User
Vendor
Wikimedia Foundation
Web request
Internet
Resolve hostname
DNS
https://howdns.works/
Get service IP
Load balancers
(LVS service IP)
Nameservers
Webrequest log
Kafka
EventLogging
Logstash
SSL
(NGINX)
Application servers
Apache
MediaWiki
(PHP/HHVM)
Local store
(APC)
Local cluster cache
(Memcached)
Metrics
statsd-lb
statsite
Graphite
HTTP Caching
Varnish
frontends
Varnish
backends
Slave databases
(MariaDB)
Primary cluster (master)
Job queue
(Redis)
Master databases
(MariaDB)
UDP2IRC
RCFeed
(Redis)

Persistency

- **Persistence** can be configured on a virtual server
  - once a server is selected, subsequent requests from the client are directed to the same server.
- Persistence is sometimes necessary in applications where client state is maintained on the server
  - but the use of persistence can cause problems in failure and other situations.
- A more common method of managing persistence is to **store state information in a shared database**
  - can be accessed by all real servers
  - A (HTTP) cookie links the client to the session.

Persistency and Fail-over

- **Failure of a service instance**: the load balancer continues to perform load balancing across the remaining services that are active.
  - Will periodically check the availability of failed service
  - One user **may have** an error returned

- **Failure of all the servers bound to a virtual server**:
  - requests may be sent to a backup virtual server (if configured)
  - optionally redirected to a configured URL (e.g. 404)

Load-Balancing Algorithms

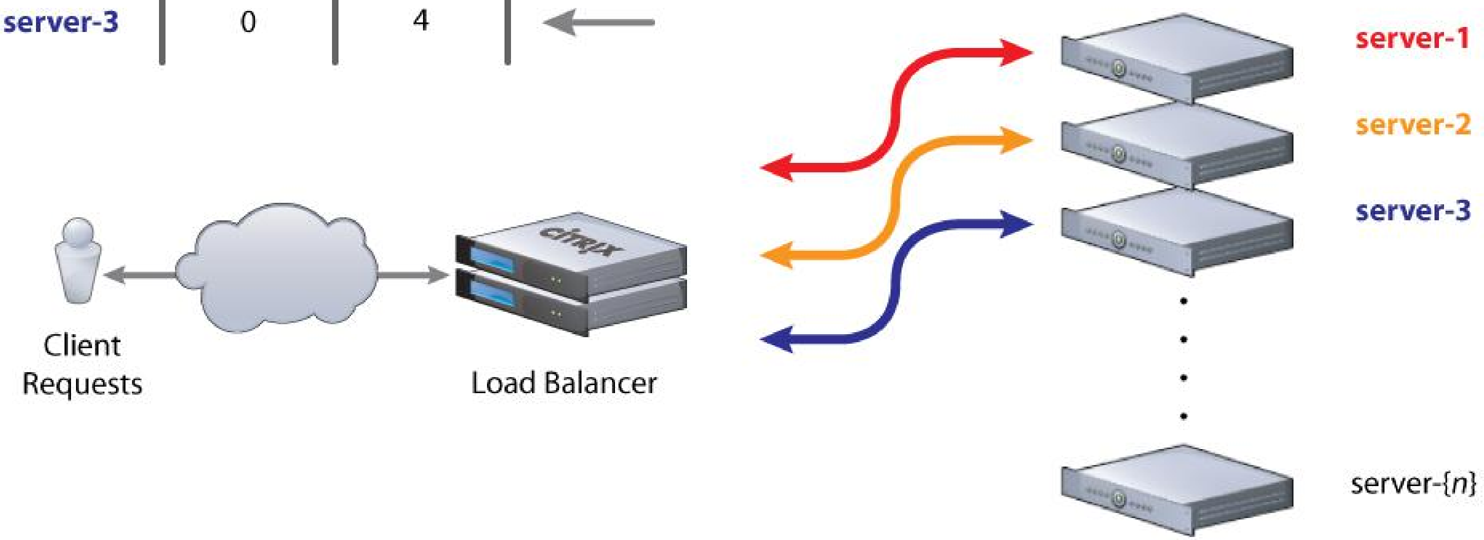
- **least connections**: server with fewest number of flows gets the new flow request.
- **weighted least connections**: associate a weight / strength for each server and distribute load across server farm based on the weights of all servers in the farm.
- **round robin**: round robin thru the servers in server farm.
- **weighted round robin**: give each server ‘weight’ number of flows in a row; weight is set just like it is in weighted least flows.
- There are other algorithms that look at or try to predict server load in determining the load of the real server.

Load-Balancing Algorithms

Load Balancing
Least Connections - Weighted

Server	Active HTTP Transactions	Weight	Next Request Serviced by
server-1	3	2	
server-2	15	3	
server-3	0	4	

N{w} = (Number of active transactions) * (10000 / weight)



Load balancing hierarchy

- Common to chain multiple balancers
  - From simplest to more complex
    - Simplest: faster but less context
    - More Complex: slower but with more context
- Common Hierarchy:
  - DNS: provide different addresses to subsequent DNS queries or addresses driven by user location
  - TCP Session: distribute clients randomly to cluster
  - TCP Port: distribute clients to specific services by port
  - Application: distribute requests based on cookie, path, other

Load balancing hierarchy

- Common Hierarchy:
  - **DNS**: provide different addresses to subsequent DNS queries or addresses driven by user location
  - **IP**: distributes users based on source/dest. address
  - **TCP Session**: distribute clients randomly to cluster
  - **TCP Port**: distribute clients to specific services by port
  - **Application**: distribute requests based on cookie, path, other

DNS - Name Resolution Balancing

- Can be implemented at a local or global scale
- Involves communication with DNS servers used by clients

- **Local**: address returned points to address of instance, mostly in a round robin approach

- **Global**: address returned points to closest entry point, mostly for reducing latency

IP - SLB: Server Load-balancing

- Gets user to needed resource without interference:
  - If user must get to same resource over and over, the SLB device must ensure that happens (ie, session persistence)
- In order to do work, SLB must:
  - Know servers – IP/port, availability
  - Understand details of some protocols (e.g., FTP, SIP, etc)
- Network Address Translation, NAT:
  - Packets are re-written as they pass through SLB device.

IP - How SLB Devices Make Decisions

- Decisions based on several factors.
  - Factors obtained from the **packet headers** (i.e., IP address, port numbers, etc.).
  - Factors obtained by looking at the data. Examples:
    - HTTP Cookies
    - HTTP URLs
    - SSL Client certificate

- The decisions can be based strictly on flow counts or they can be based on knowledge of application.

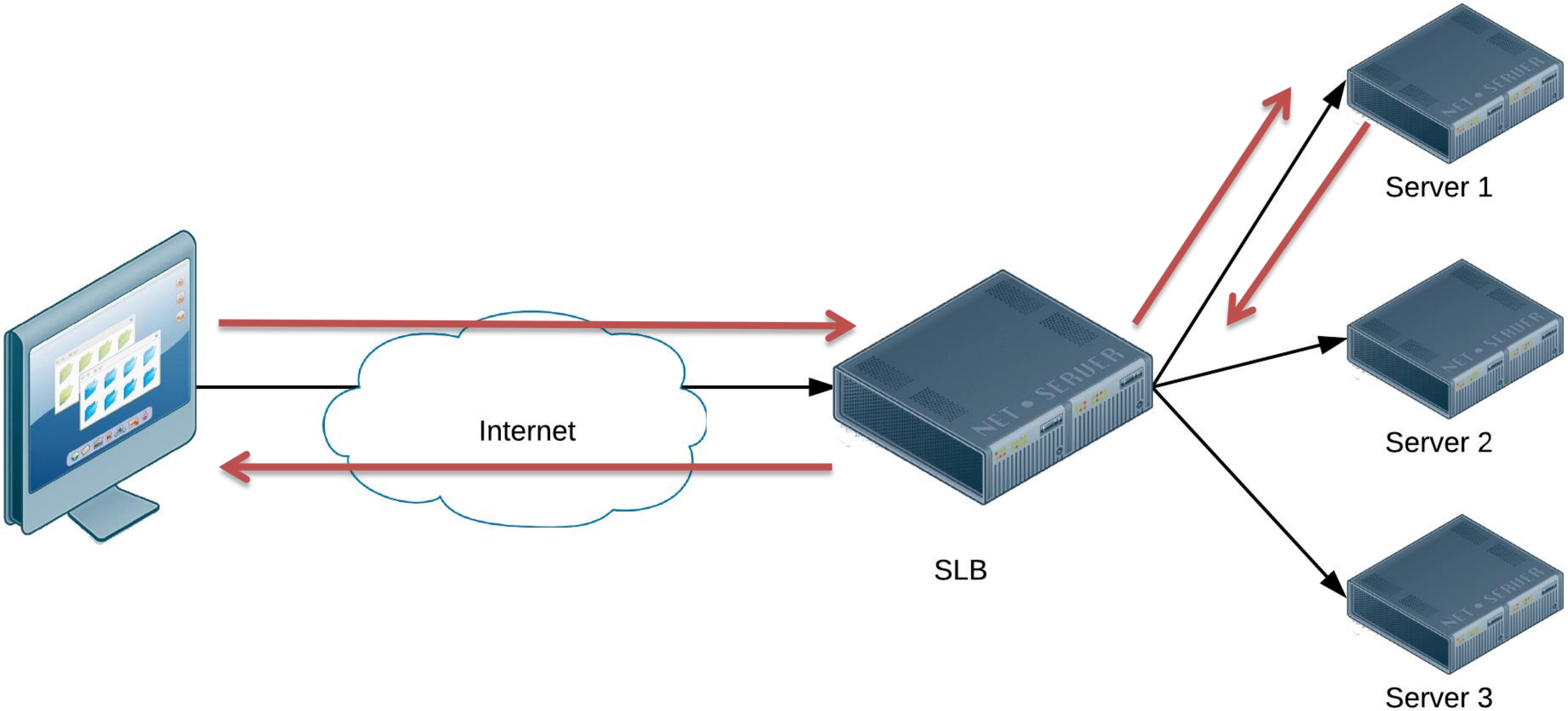
- For some protocols, like FTP, you have to have knowledge of protocol to correctly load-balance (i.e., control and data connection must go to same physical server).

IP - SLB Operation

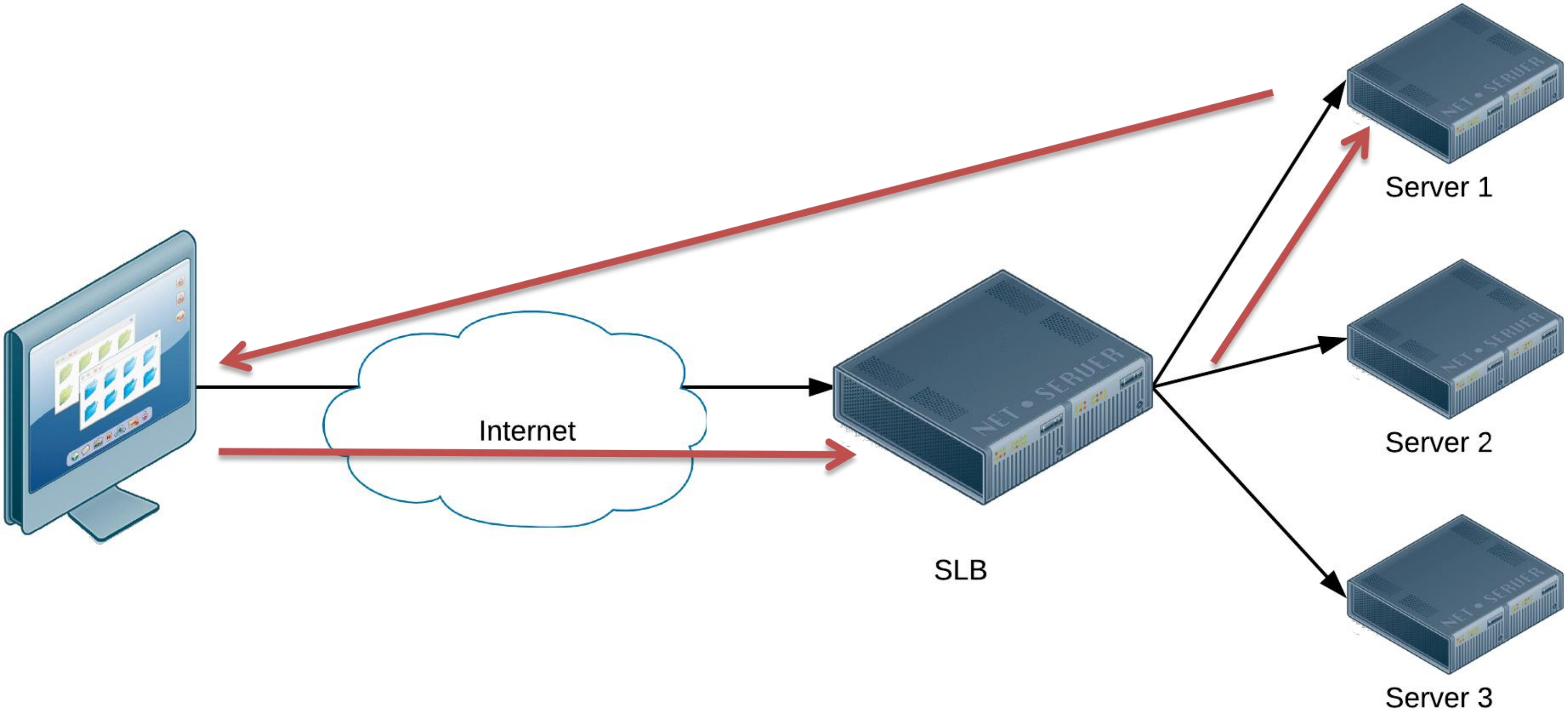
- On a new flow, it determines if the virtual server exists.
  - If so, make sure virtual server has available resources.
  - If so, then determine level of service needed by that client to that virtual server.
    - If virtual machine is configured with particular type of protocol support of session persistence, then do that work.
  - Pick a real server for that client.
    - The determination of real server is based on flow counts and information about the flow.
    - In order to do this, the SLB may need to proxy the flow to get all necessary information for determining the real server – this will be based on the services configured for that virtual server.

- If not, the packet is bridged to the correct interface based on Layer 2.

IP - SLB with NAT



IP - SLB without NAT



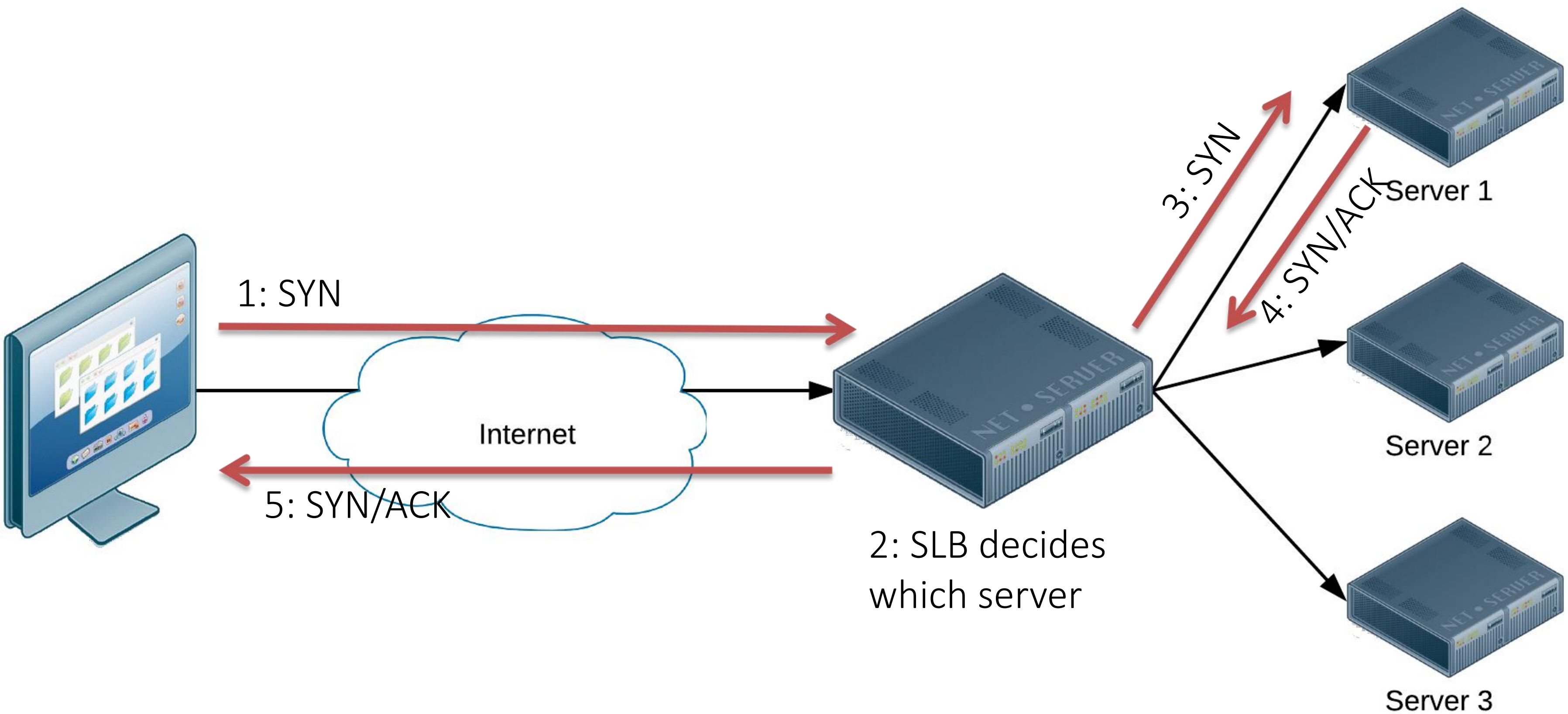
TCP - Load-Balance

- Looking at the destination IP address and port to make a load-balancing decision.

- In order to do that, it can determine a real server based on the first packet that arrives.

- Objectives:
  - Distribute load: round robin through Equal Cost Multi Path
  - Dispatch to specific hosts: port based

TCP - SLB @ layer 3/4

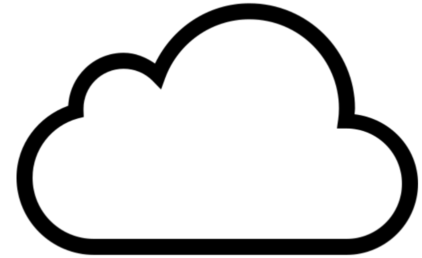


TCP - ECMP

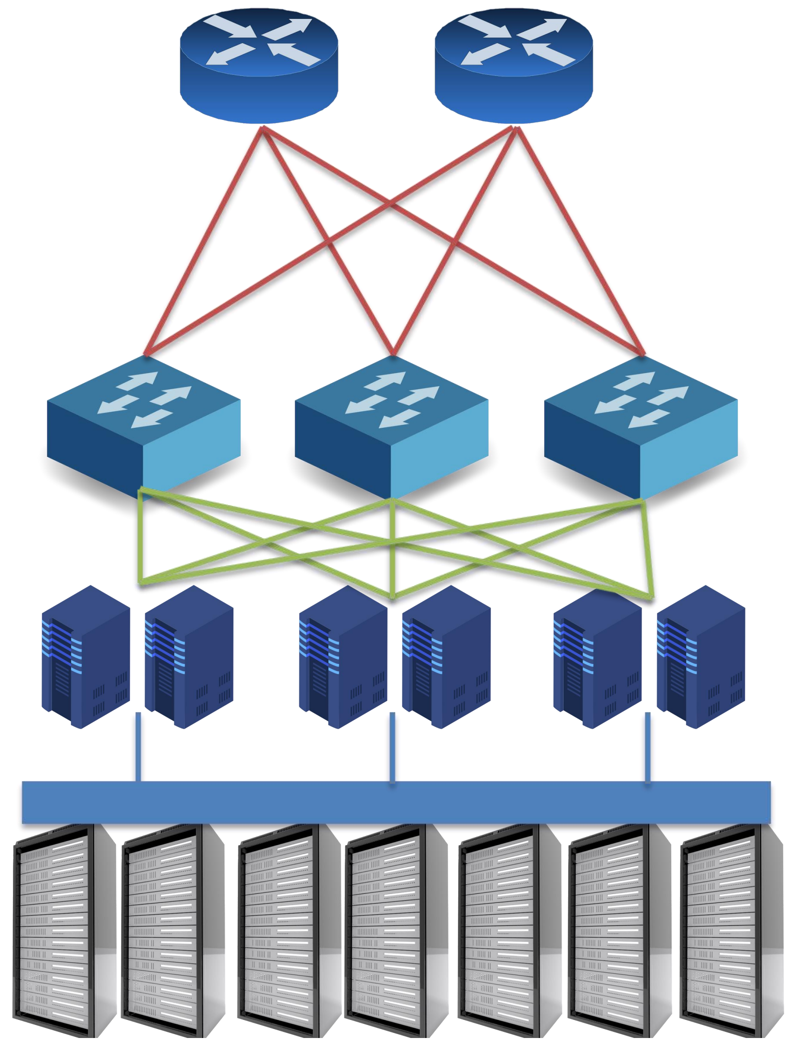
- Considers that the entry router **may have multiple routes** to the same IP address
  - These IP addresses may be in different servers!
- Sends packets through different routes
  - In the basic form, it’s a problem for stateful traffic (TCP)

- Resilient ECMP
  - Router calculates hash based on 5 parameters: IP Proto, IP src/dst, TCP src/dst
  - Based on the hash, it routes packets through a route
    - all packets from same session will go through the same route

TCP - ECMP

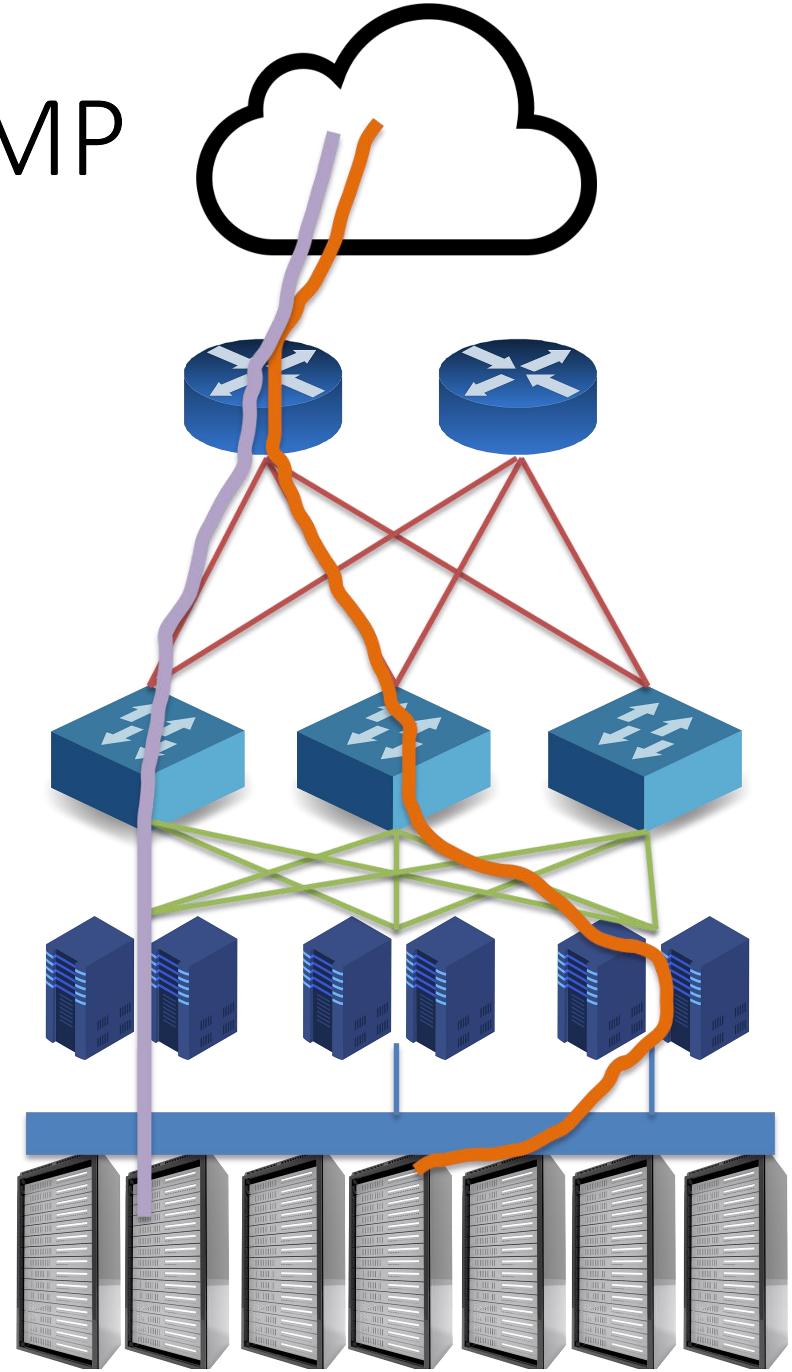


- VLANs, and other segmenting is possible
- Router typically dispatchs flow to higher layer LB
- Problems:
  - Hash colision
  - Link saturation
    - esp. w/ assymetric links



TCP - ECMP

- VLANs, and other segmenting is possible
- Router typically dispatchs flow to higher layer LB
- Problems:
  - Hash colision
  - Link saturation
    - esp. w/ assymetric links

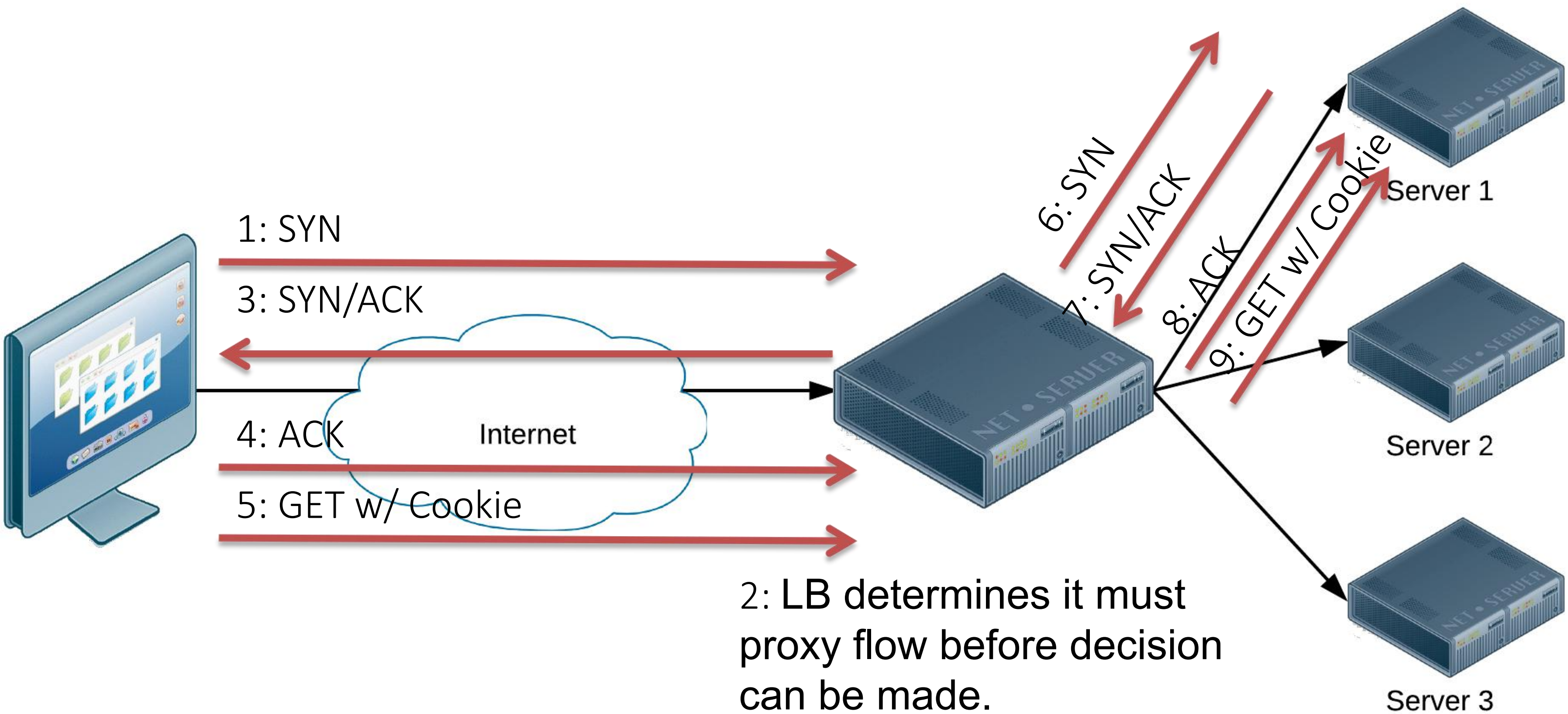


Layer 5+: Load-Balance

- The LB must terminate the TCP flow for **BEFORE** the decision can be made.
  - Eg: the cookie must be sent by the client,
    - which is after the TCP handshake and before determining the real server.

- This also applies to TLS
  - uses SNI: Server Name Indication
  - Provides users with the correct certificate

SLB @ layer 5+



Rest of flow continues with Server response.
Note: the flow can be unproxied at this point for efficiency.

Server Feedback

- LB may need information from the real servers while they are part of the server farm
  - Or of the used resources (storage, network)

- Why?
  - Dynamic decisions based on ability of real server.
  - Dynamic provisioning of applications.
  - Allow maintenance without downtime
  - Avoid link saturation

- Why not?
  - Complexity, which leads to lower performance

Server Feedback: Use of Information

- To determine health of real servers, LB can:
  - **Actively monitor flows** to that real server.
  - **Initiate probes** to the real server.
  - **Get feedback** from real server or third party box.
- Availability of real server is reported as a ‘weight’ that is use by LB algorithms
- As weight value changes over time, the load distribution changes with it.

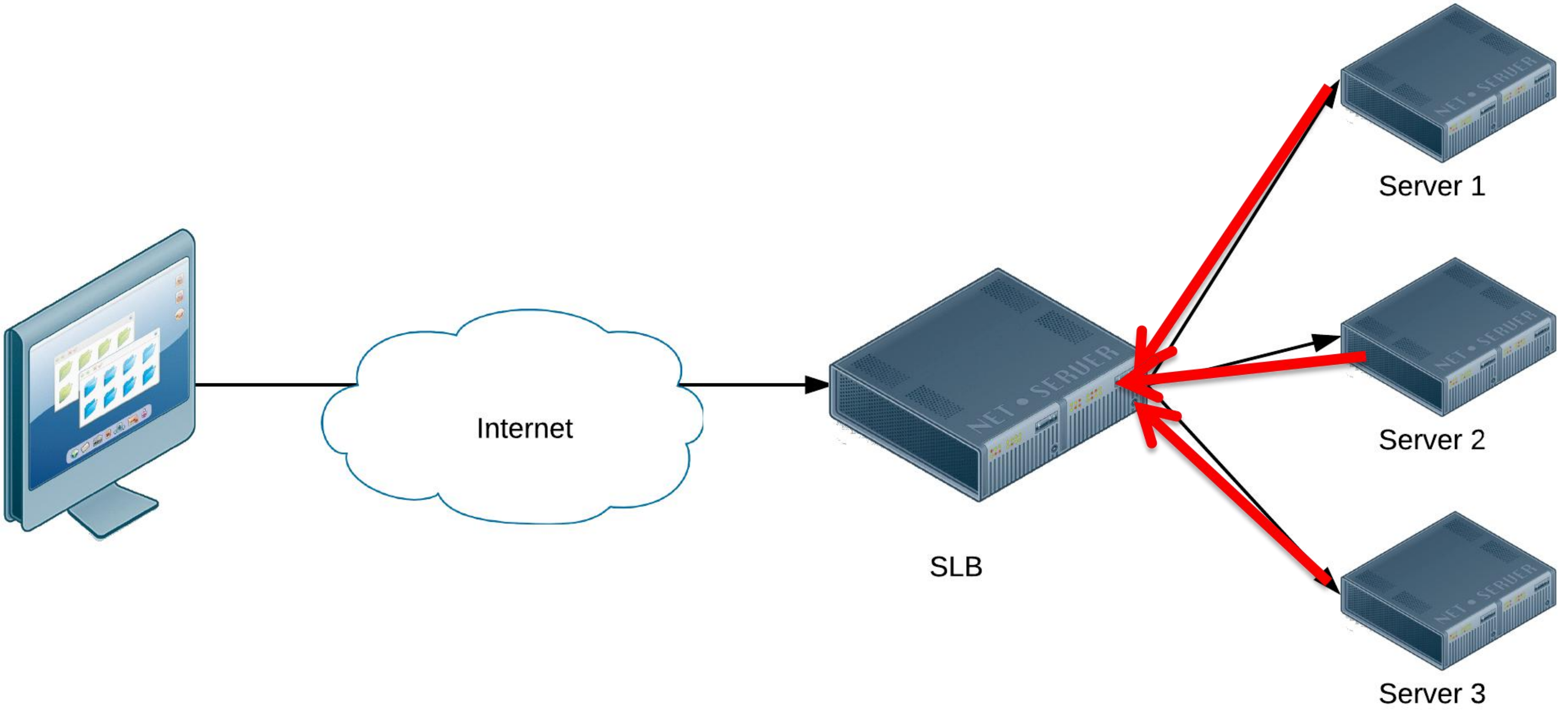
How to Get Weights

- Statically configured on LB – never change.
- Start with statically configured value on LB device for initial start-up, then get weight from:
  - Real server
  - Third party box / Collection Point
    - It is assumed that if a third party box is being used, it would be used for all the real servers in a server farm.

Direct Host Feedback

- Have “agents” running on host to gather data points.
- Data is then sent to LB device just for that physical server.
  - Note: agent could report for different applications on that real server.
  - Agent could be based on available memory, general resources available, proprietary information, etc.

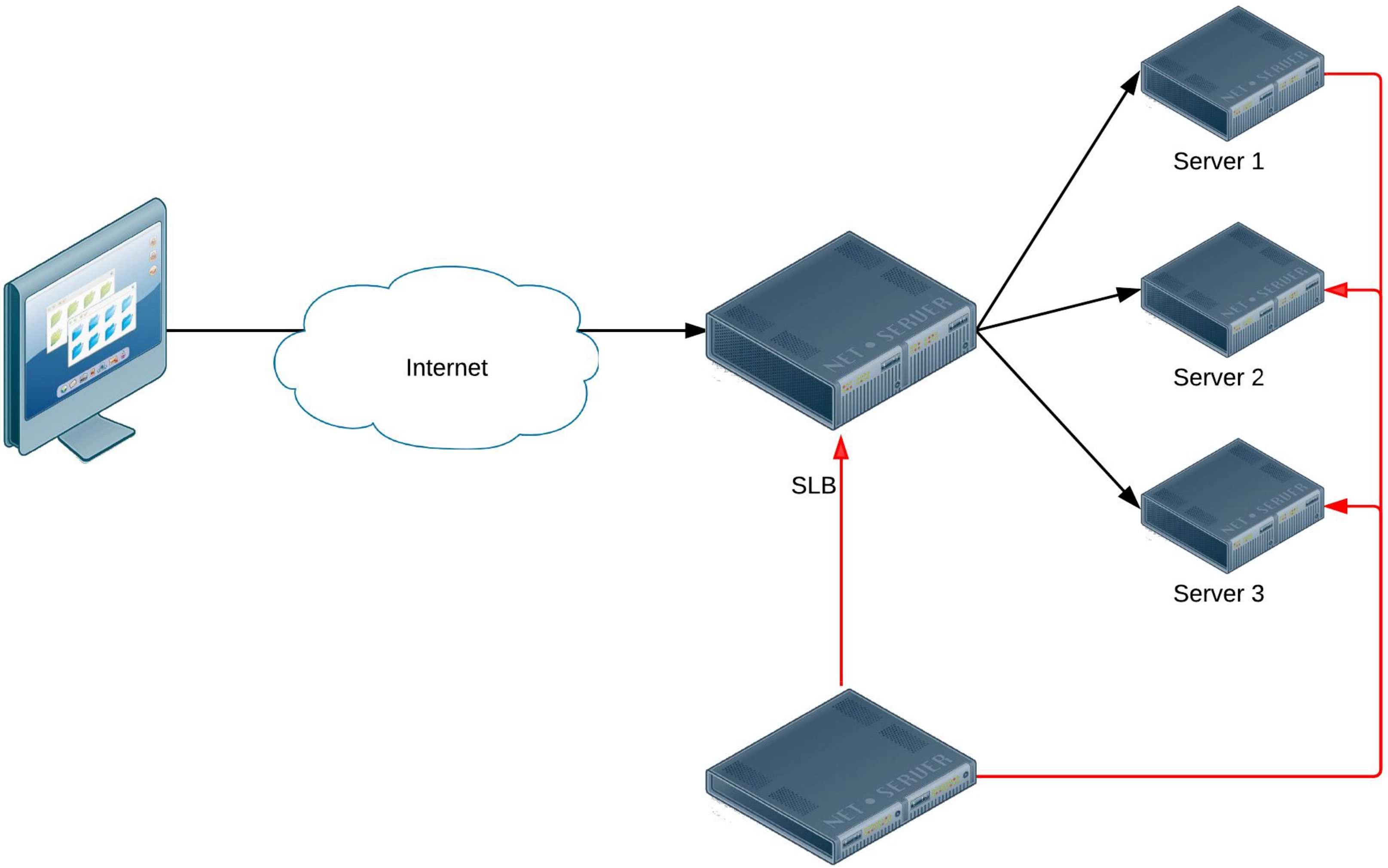
Direct Feedback



Direct Host Feedback

- Pros:
  - Have some way to dynamically change physical server’s capability for LB flows.
- Cons:
  - LB must attempt to normalize data for all real servers in a server farm. If have heterogeneous server
    - it is difficult to do so
  - Difficult for real server to identify itself in LB terms for case of L3 vs. L4 vs. L5, etc LB scenarios.

Third Party Feedback: Network



Host to Third Party Feedback

- Real servers report data to a ‘collection point’.
  - The ‘collection point’ system can normalize the data as needed, then it can report for all physical servers to the LB.
  - Or the collection point **pools** servers!

- Can reuse monitoring infrastructure
  - Nagios, Cacti, other SNMP, etc…

Host to Third Party Feedback

- Pros:
  - Have a device that can analyze and normalize the data from multiple servers. The LB can then just do LB functionality.

- Cons:
  - Requires more communication to determine dynamic weight – could delay the overall dynamic affect if it takes too long.

Web Server Load Balancing

- One major issue for large Internet sites is how to handle the load of the large number of visitors they get.
- This is routinely encountered as a scalability problem as a site grows.
- There are several ways to accomplish load balancing

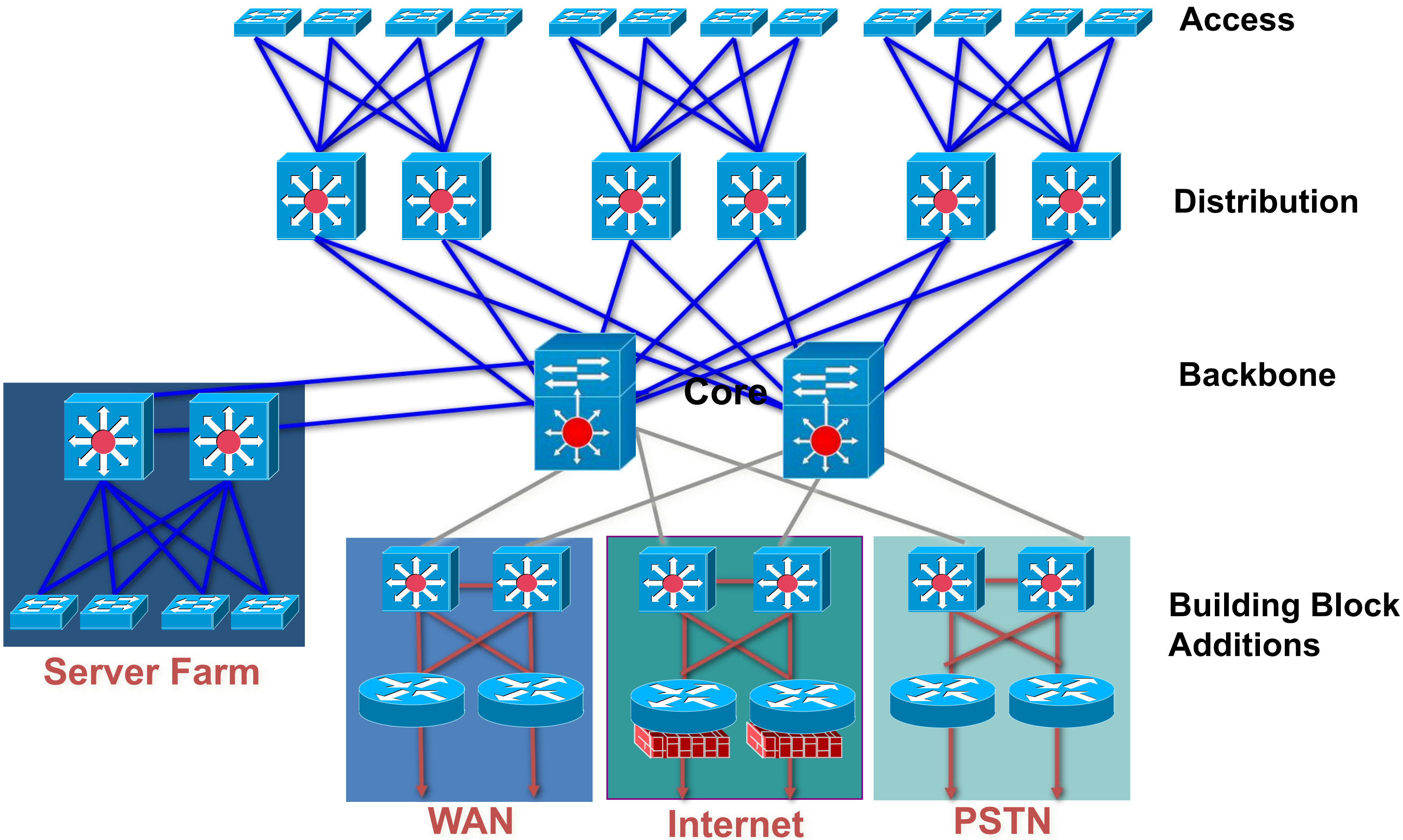
Network Load Balancing

- Use Cases
  - Failover
  - Multiple ISP providers
  - Channel bounding & Layer 2 & Layer 3 load-balancing using Network Equipment's (Switches and Routers)

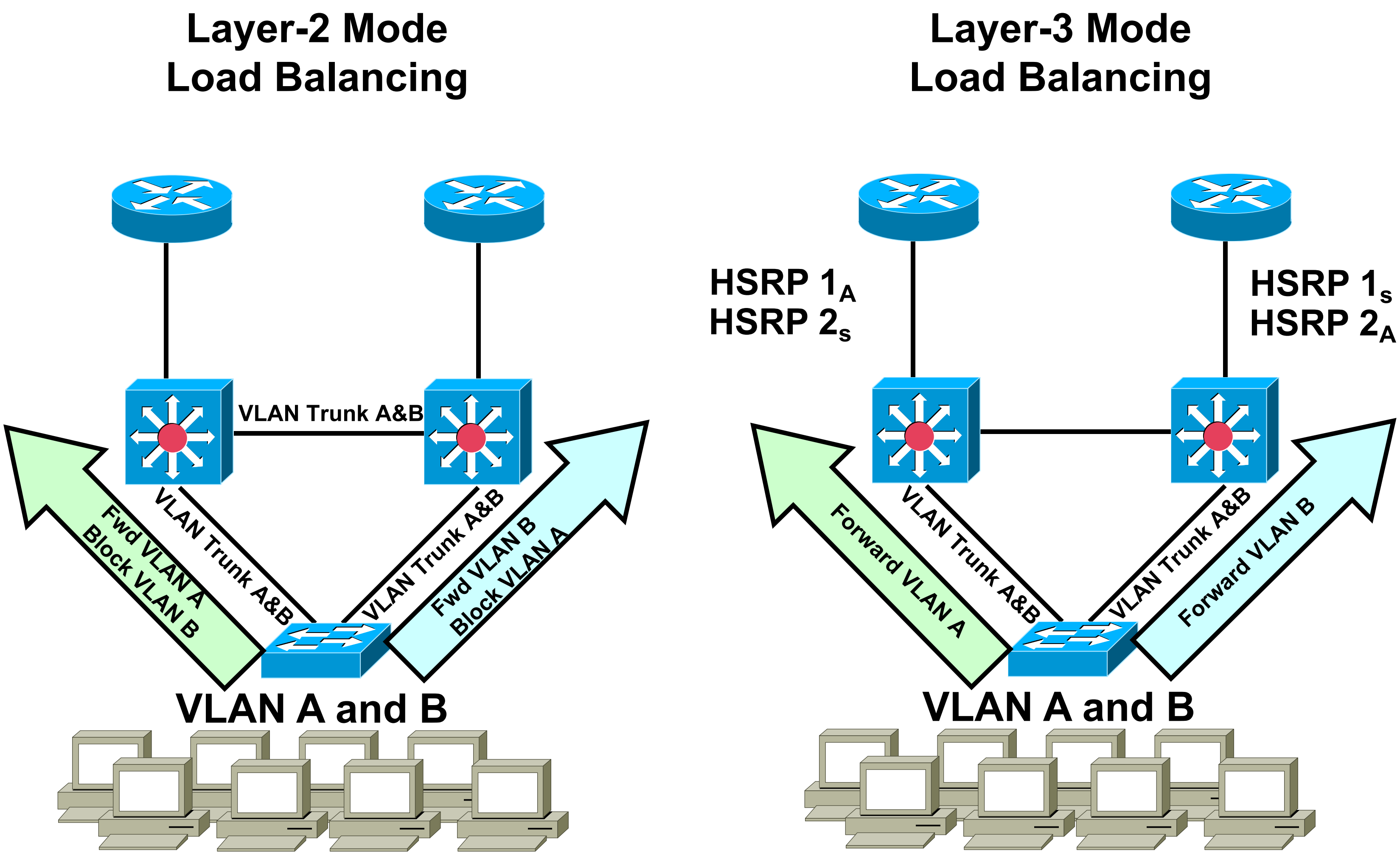
- Layer 2
  - Mostly using spanning tree protocol or OpenFlow

- Layer 3
  - HSRP, VRRP, GLBP

Multilayer Network Design



Multi-VLAN Load Balancing Methods



First Hop Redundancy Schemes

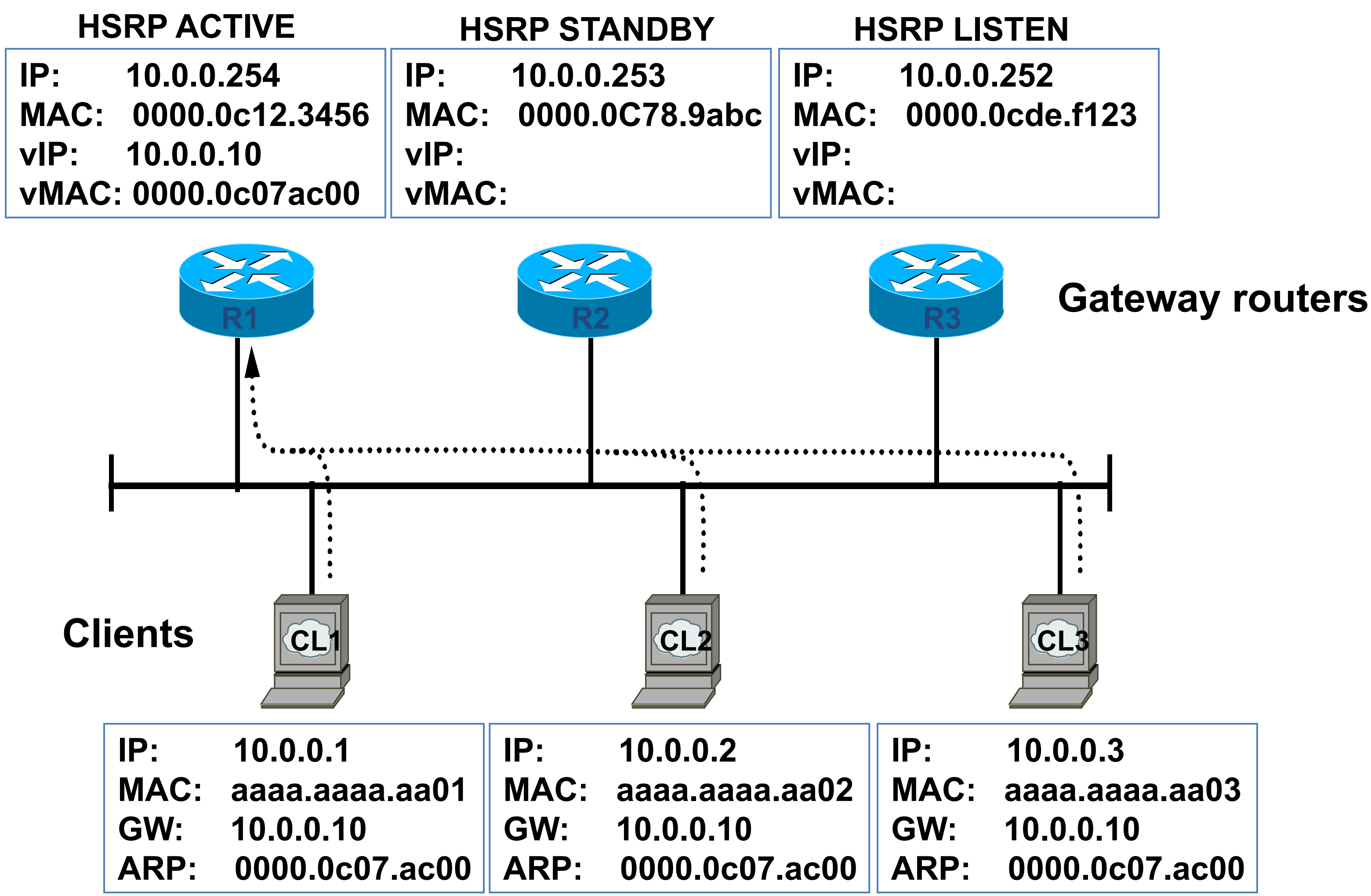
- Hot Standby Router Protocol (HSRP)
  - Cisco informational RFC 2281 (March 1998)
- Virtual Router Redundancy Protocol (VRRP)
  - IETF Standard RFC 2338 (April 1998)
- Gateway Load Balancing Protocol (GLBP)
  - Cisco designed, load sharing, patented

HSRP

- A group of routers function as **one virtual router** by sharing ONE virtual IP address and ONE virtual MAC address
- One (Active) router performs packet forwarding for local hosts
- One router provides “hot standby” in case the active router fails
- Standby routers stay idle as far as packet forwarding from the client side is concerned

First Hop Redundancy with HSRP

R1- Active, forwarding traffic; R2, R3 - hot standby, idle



VRRP

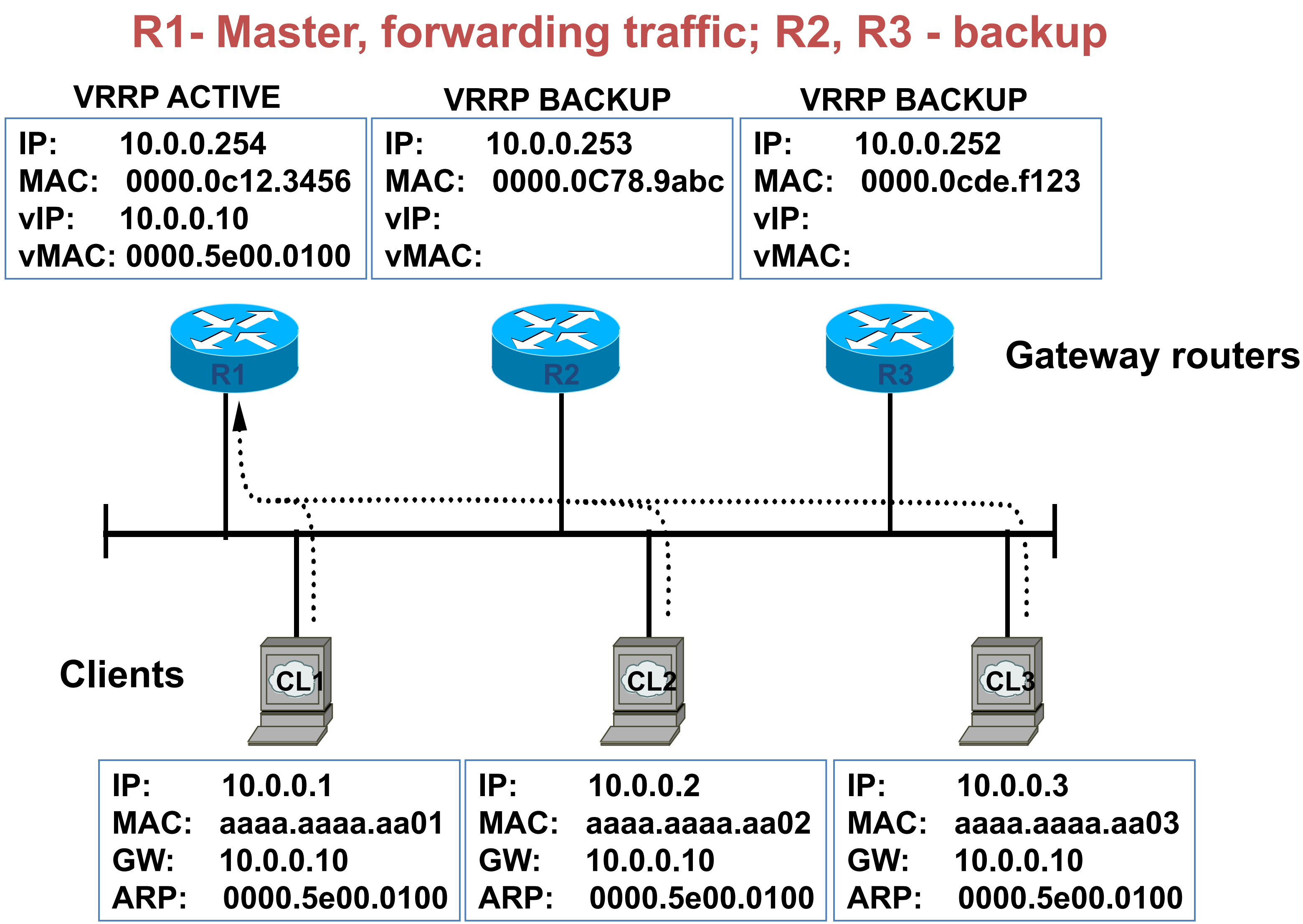
- A group of routers function as **one virtual router** by sharing ONE virtual IP address and ONE virtual MAC address

- One (master) router performs packet forwarding for local hosts

- The rest of the routers act as “back up” in case the master router fails

- Backup routers stay idle as far as packet forwarding from the client side is concerned

First Hop Redundancy with VRRP



GLBP Defined

- A group of routers function as one virtual router by sharing ONE virtual IP address but using MULTIPLE virtual MAC addresses for traffic forwarding

- Provides uplink load-balancing as well as first hop fail-over

GLBP Requirements

- Allow traffic from a single common subnet to go through multiple redundant gateways using a single virtual IP address
- Provide upstream load-balancing by utilizing the redundant up-links simultaneously
- Eliminate the need to create multiple vLANs or manually divide clients for multiple gateway IP address assignment
- Preserve the same level of first-hop failure recovery capability as provided by HSRP

First Hop Redundancy with GLBP

R1- AVG; R1, R2, R3 all forward traffic