High Availability

Gestão de Infraestruturas de Computação

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

João Paulo Barraca

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)**

- **Availability estimation and measurement**
  - MTBF / (MTBF + MTTR)
    - Mean Time Between Failures
    - Mean Time To Recover
  - (Total Uptime / Total Time) * 100%

Definitions

High Availability

- **High availability (HA) is a characteristic of a system that aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period**

- **Increasingly important for current Internet services**
  - High impact to brand
  - Small tolerance to failures by users
  - Wide scale of internet can cause fast emergence of traffic spikes, leading to failure
  - Systems are increasingly complex, with more points of failure

High Availability

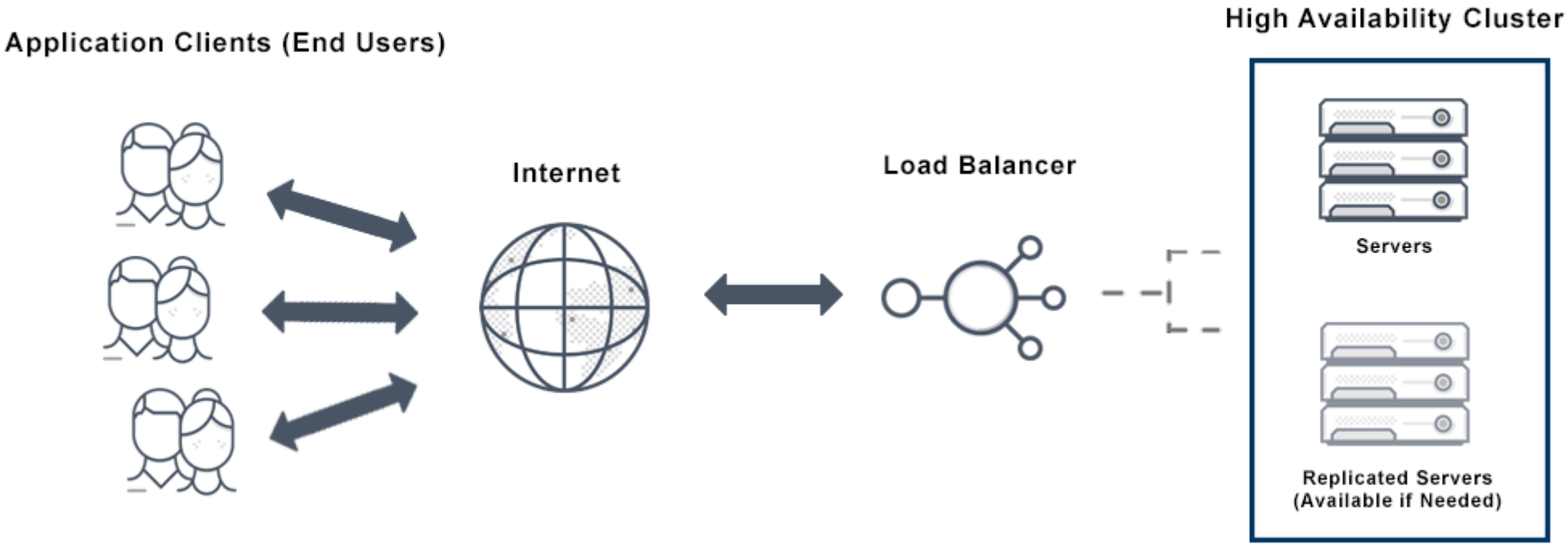


Image source: Avi Networks

Availability vs Reliability

- **Availability refers to the probability of a service being provided in the future**
  - that is, the servers are operational and will provide an answer to a request
- **Reliability refers to the client actually getting the service requested**
  - that is, receiving a correct response, and not an error
- **High availability, low reliability: service is provided by with low quality**
  - corrupted messages, long delays, errors
- **Low availability, high reliability: service fails frequenty but clients never use the service when it fails**
  - it fails outside office times

High Availability

Design Principles

- **Eliminate single points of failure**
  - No single system, data store, entity, endpoint, network connection, software instance
- **Reliable crossovers**
  - Ability of a system to switch from one component to another without losing any data or reducing performance
- **Failure detection**
  - Observe system operation and detect failures
  - May involve active probing systems

Eliminate single points of failure

- **Components must have additional elements to handle individual failures**
  - Software, server, data center site

- **Types**
  - Software
  - Network
  - Geography
  - Server
  - Data

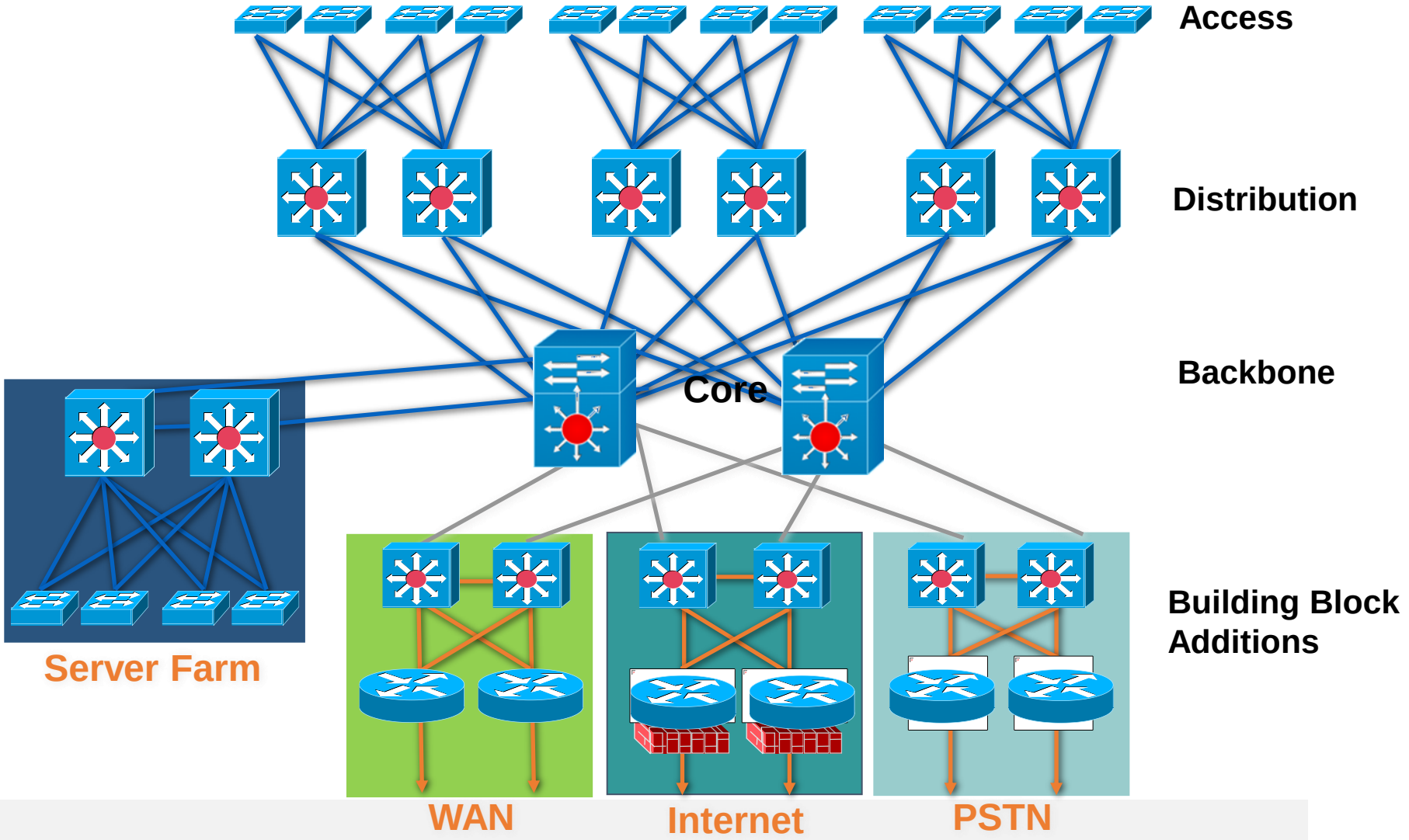
- **Common arrangements**
  - N+1 – One surplus equipment to handle one failure
  - 2N – Double the capacity to handle failure of half the elements
  - 2(N+1) – As before but local systems have one additional redundancy

Redundant systems

Photo: Rainer Knäpper, Free Art License

Image Source: Clayton Industries

Redundant systems



Redundant systems

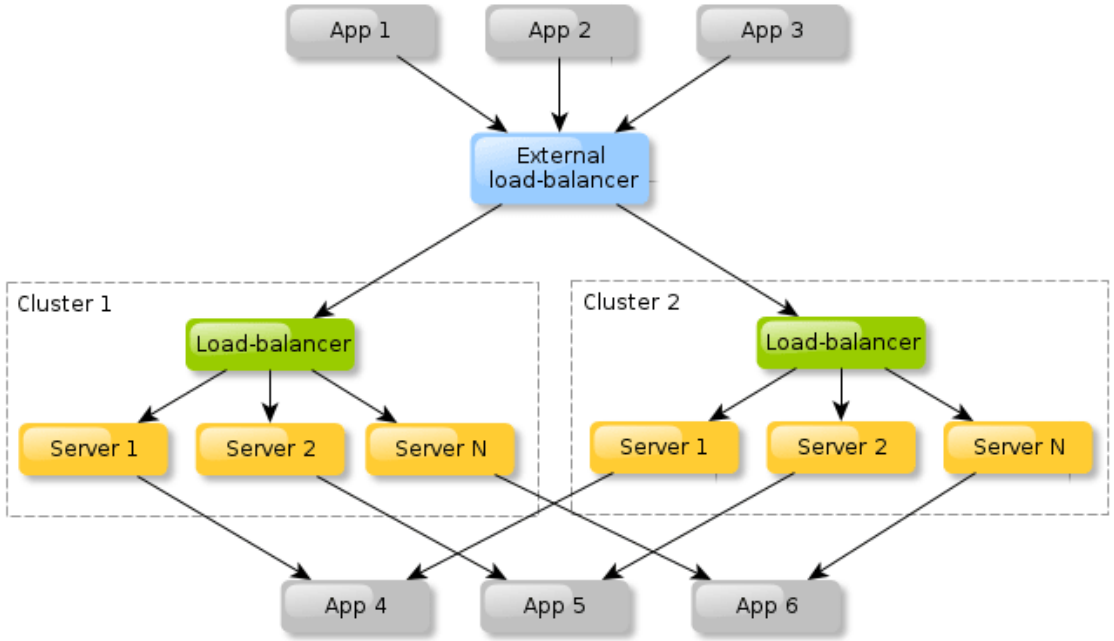


Image source: liquidweb.com

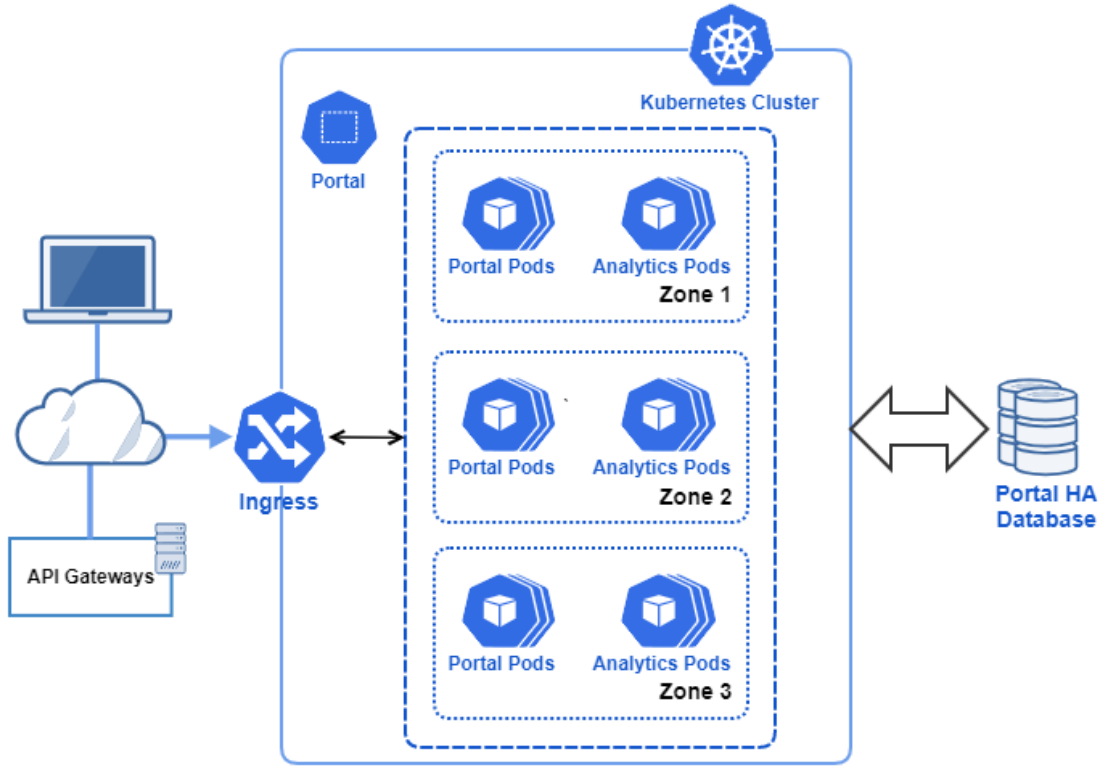
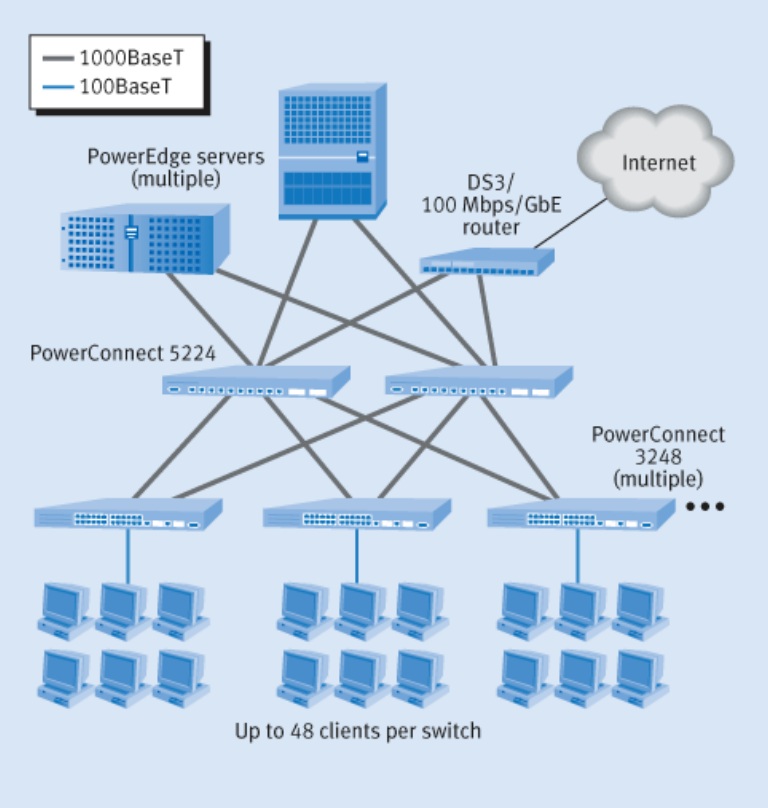


Image source: broadcom.com

Reliable crossover

- **Property related to the swift transition between systems**
  - No (or reduced) service loss or degradation with transition
  - Examples
    - Very fast transition between power sources or links
    - Keeping shared session state



Failure detection

- **Set of mechanisms and processes able to detect issues in service operation**
  - Implemented both at small and large scale
    - Small scale: detect Ethernet link failure
    - Large scale: detect failure of Cloud zone
  - Trigger recovery mechanisms

- **Inherently technology dependent**
  - Analyse logs/packet rate/errors
  - Track Network link state
  - Track TCP sessions
  - Proactivelly probe systems
    - HTTP queries, ICMP packets, TCP connections
  - Keep alive / Beacons

Definition

Service Level Agreements

- **An SLA is an agreement between the service provider and the client**
  - It frames the service provided to the client under some expectations and obligations
  - There may be clauses for failure to comply the SLA and compensations
  - The client may be the service provider itself
- **Validating an SLA requires monitoring SLOs under the scope of SLIs**
  - Service Level Object is a metric to be observed
  - Service Level Indication is a threshold or function applied to a esult

Definitions

Service Level Agreements

- **SLA Includes**
  - Where the Service Works
  - Responsibilities of the Service
  - Warranty of the Service
  - Guarantee of the Service
  - Ease and Performance of the Service
  - Customer Support

Definitions

Service Level Agreements

- **In computation infrastructures there is a wide range os SLOs. Some are:**
  - **Capacity**: Amount of persistent/volatile memory
  - **Latency**: Time it takes to complete an operation
  - **Throughput**: Amount of data transfered per unit of time
  - **Availability**: Percentage of time the service is provided
  - **Recovery time**: Time it takes to recover from failure
  - **Reliability**: Probability of a client request be correct

- **SLIs mostly refer to thresholds and averages over the SLOs observed**
  - ex: Max latency, min and max throughput
  - SLIs can indicate an SLA failure or just a warning state
  - The structure may be rich: 90% of the time, latency should be below 100ms

Definitions

Downtime

- **Disruptions causing the system to not provide service**
  - They always exist and cannot be completelly avoided

- **Unplanned downtime: due to failures**
  - Physical failures: HVAC, hardware, power loss, human error
  - Software failures: error in code, misconfiguration, cyberattack

- **Planned downtime: due to maintenance updates and optimization**
  - New software versions
  - Updating base systems
  - Migrating to diferent servers, upgrade network gear

Reliability in TIA-492

- **TIA-492 defines guidelines for computational infrastructures**
  - Cabling, redundancy, pathways, construction guidelines, etc..
- **Also defines benchmarks for Tiered Reliabiliy**
  - Each Tier provides higher reliability
  - Each tier have requirements on architecture, components
  - Each tier targets an Availability value
- **Most computational systems follow these tiers, even when not implementing the TIA-492 standard**

Reliability in TIA-492

- **Tier I - Basic: 99.671% Availability**
  - Susceptible to disruptions from both planned and unplanned activity
  - Single path for power and cooling, without redundant components
  - Anual downtime of 28.8 hours
  - Preventime maintenance requires system shutdown
    - Impacts availability, but may not impact perceived availability

- **Tier II - Redundant: 99.741% Availability**
  - Less susceptible to disruptions from both planned and unplanned activity
  - Single path for power and cooling with redundant components (N+1)
  - Includes Raised Floor, UPS, Generator
  - Anual downtime of 22.0 hours
  - Maintenance of power paths require some form of shutdown

Reliability in TIA-492

- **Tier III - Concurrently Maintainable: 99.982 Availability**
  - Planned activities do not disrupt the system, but unplanned activities may
  - Multiple power and cooling paths, with only one active, includes redundant components (N+1)
  - Anual downtime of 1.6 hours
  - Raised floor and enough capacity for maintaining half of the systems
    - Costs are at least the double of Tier I

- **Tier IV - Fault Tolerant: 99.995% Availability**
  - Planned activities do not disrupt the system, as well as one unplanned activity
  - Multiple power and cooling paths, includes redundant components (2 * (N+1), i.e. 2 UPS, each with + 1 system)
  - Anual downtime of 0.4 hours

Basic structure for internet services

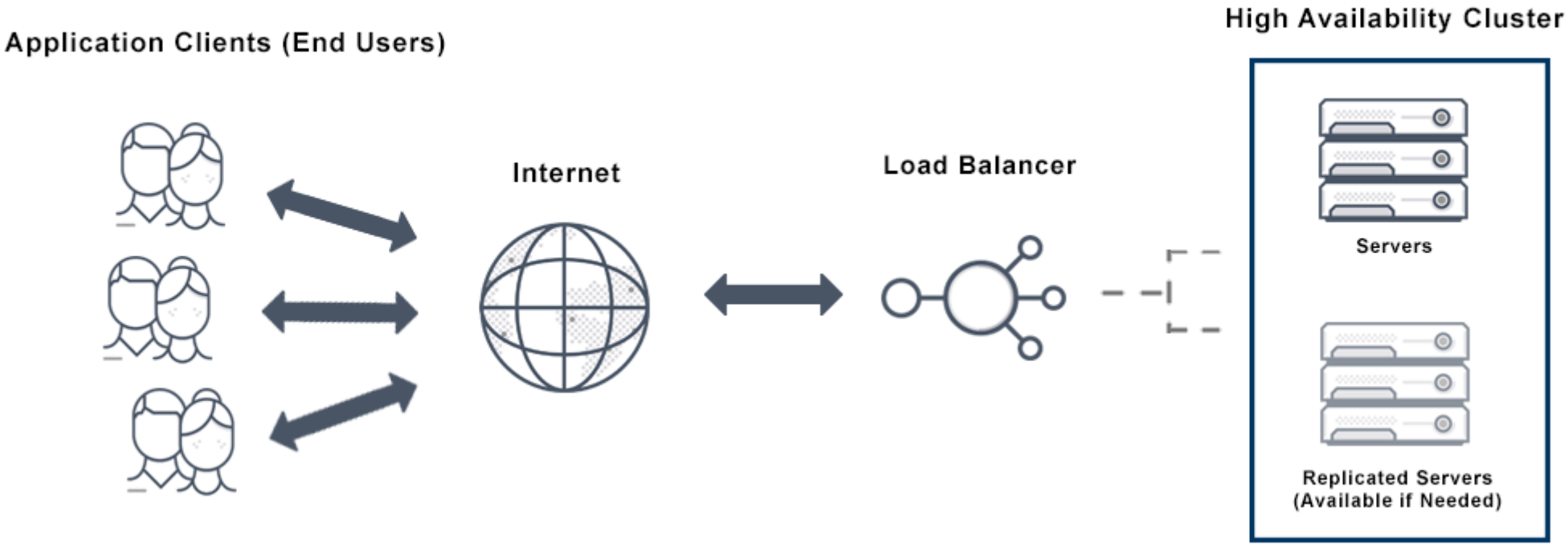
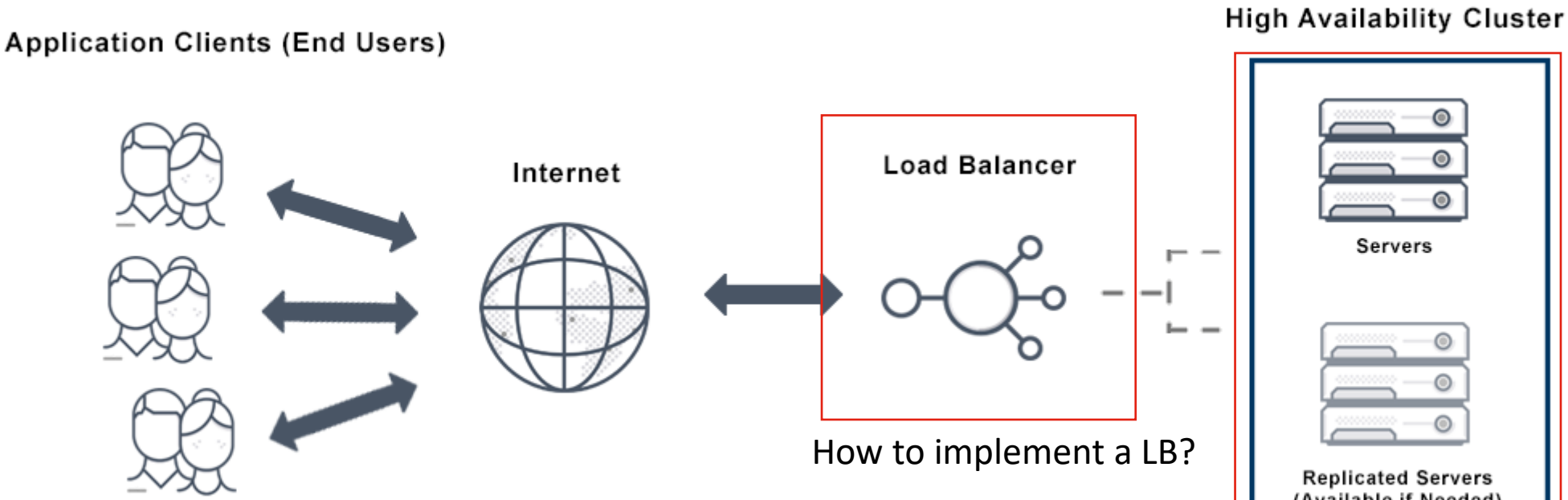


Image source: [Avi Networks]

Basic structure for internet services



How to implement a LB?

How to handle failover?

Image source: Avi Networks

Failover models

- **Active-Active**
  - All systems are providing service at the same time
    - Some state must be synchronized between them for shared transactions
  - Failure of a single system will be managed by others automatically

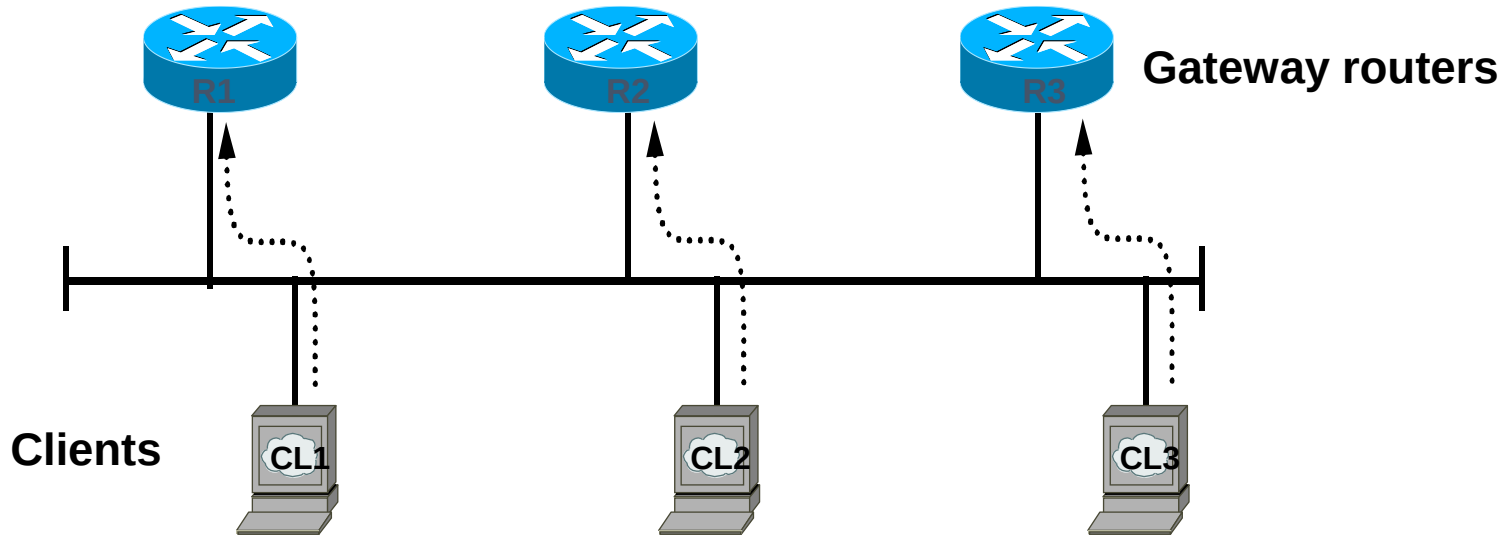
- **Active-Active with load balancer**
  - Similar to the previous but there is an active component (LB) distributing requests

- **Active-Standby**
  - One system is active, while N other are in standby
  - Failure of the active system will upgrade a standby system to active
  - Failure detected by external system or lack of feedback from active system
    - E.g. using heartbeats

GLBP: Active-Active approach

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

GLBP AVG/AVF,SVF	GLBP AVF,SVF	GLBP AVF,SVF
IP: 10.0.0.254	IP: 10.0.0.253	IP: 10.0.0.252
MAC: 0000.0c12.3456	MAC: 0000.0C78.9abc	MAC: 0000.0cde.f123
vIP: 10.0.0.10	vIP: 10.0.0.10	vIP: 10.0.0.10
vMAC: 0007.b400.0101	vMAC: 0007.b400.0102	vMAC: 0007.b400.0103

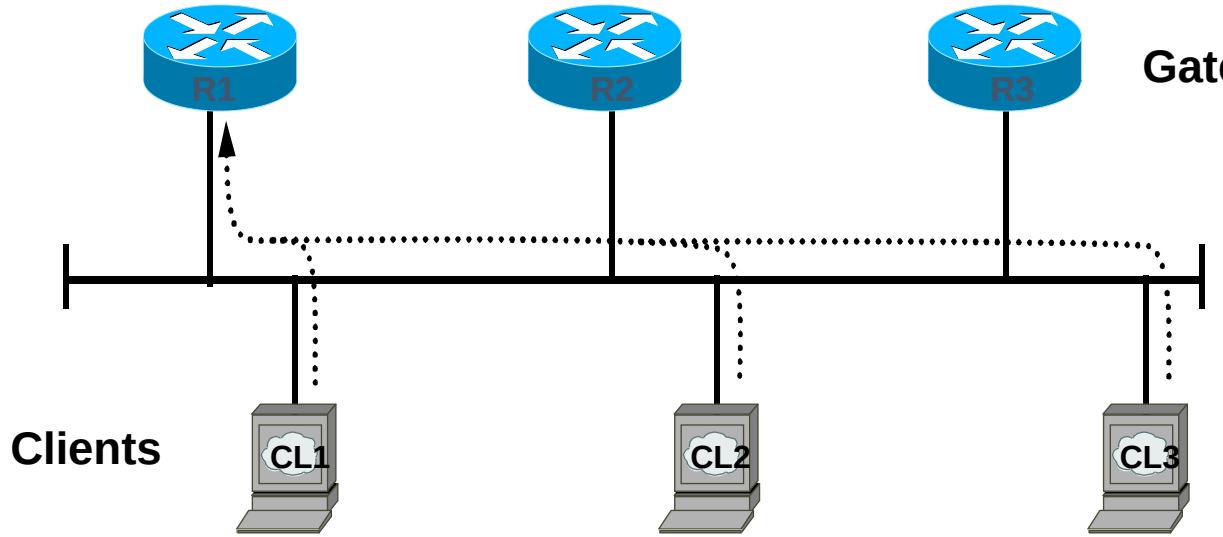


IP: 10.0.0.1	IP: 10.0.0.2	IP: 10.0.0.3
MAC: aaaa.aaaa.aa01	MAC: aaaa.aaaa.aa02	MAC: aaaa.aaaa.aa03
GW: 10.0.0.10	GW: 10.0.0.10	GW: 10.0.0.10
ARP: 0007.B400.0101	ARP: 0007.B400.0102	ARP: 0007.B400.0103

HSRP: Active-Standby approach

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

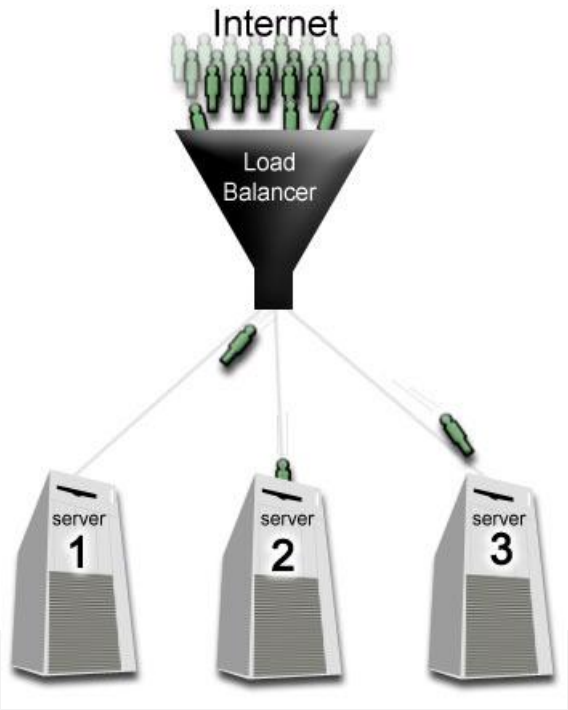
HSRP ACTIVE	HSRP STANDBY	HSRP LISTEN
IP: 10.0.0.254 MAC: 0000.0c12.3456 vIP: 10.0.0.10 vMAC: 0000.0c07ac00	IP: 10.0.0.253 MAC: 0000.0C78.9abc vIP: vMAC:	IP: 10.0.0.252 MAC: 0000.0cde.f123 vIP: vMAC:



IP: 10.0.0.1 MAC: aaaa.aaaa.aa01 GW: 10.0.0.10 ARP: 0000.0c07.ac00	IP: 10.0.0.2 MAC: aaaa.aaaa.aa02 GW: 10.0.0.10 ARP: 0000.0c07.ac00	IP: 10.0.0.3 MAC: aaaa.aaaa.aa03 GW: 10.0.0.10 ARP: 0000.0c07.ac00

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 **cluster** 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.



Virtual Server

- **Different virtual servers can be configured for different sets of physical services, such as TCP and UDP services in general**
  - Usually on a Virtual IP (Floating IP)

- **Application specific virtual server may exist to support HTTP, FTP, SSL, DNS**
  - **provided** by an additional server
  - HTTP Proxy
  - DNS forwarder
  - TLS connection terminator

- **Load balancing methods manage the selection of an appropriate server in a cluster**
  - Physical server, software instance, other lower level LB

Load Balancers

- **Load balancer maintains the system in a operational 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)

- **Common assumptions**
  - Nodes are cooperative
  - Only one bottlenecked resource

Why to Load-balance?

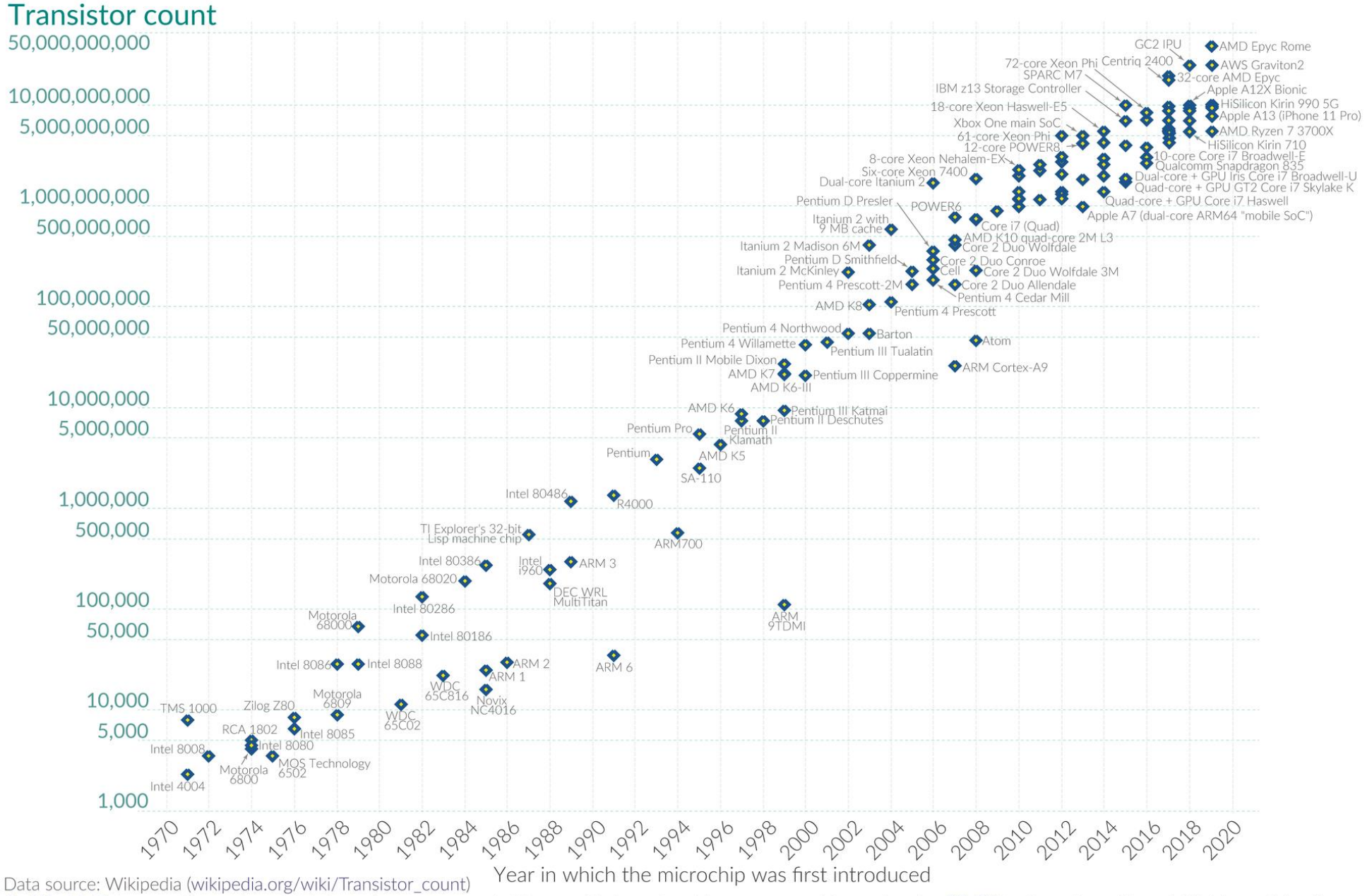
- **Time to serve request is bound to capacity of a single CPU + storage latency + database latency**
  - Dynamic Content: **Tens** of reqs/cpu/second
  - Static Content: **Thousands** of reqs/cpu/second

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

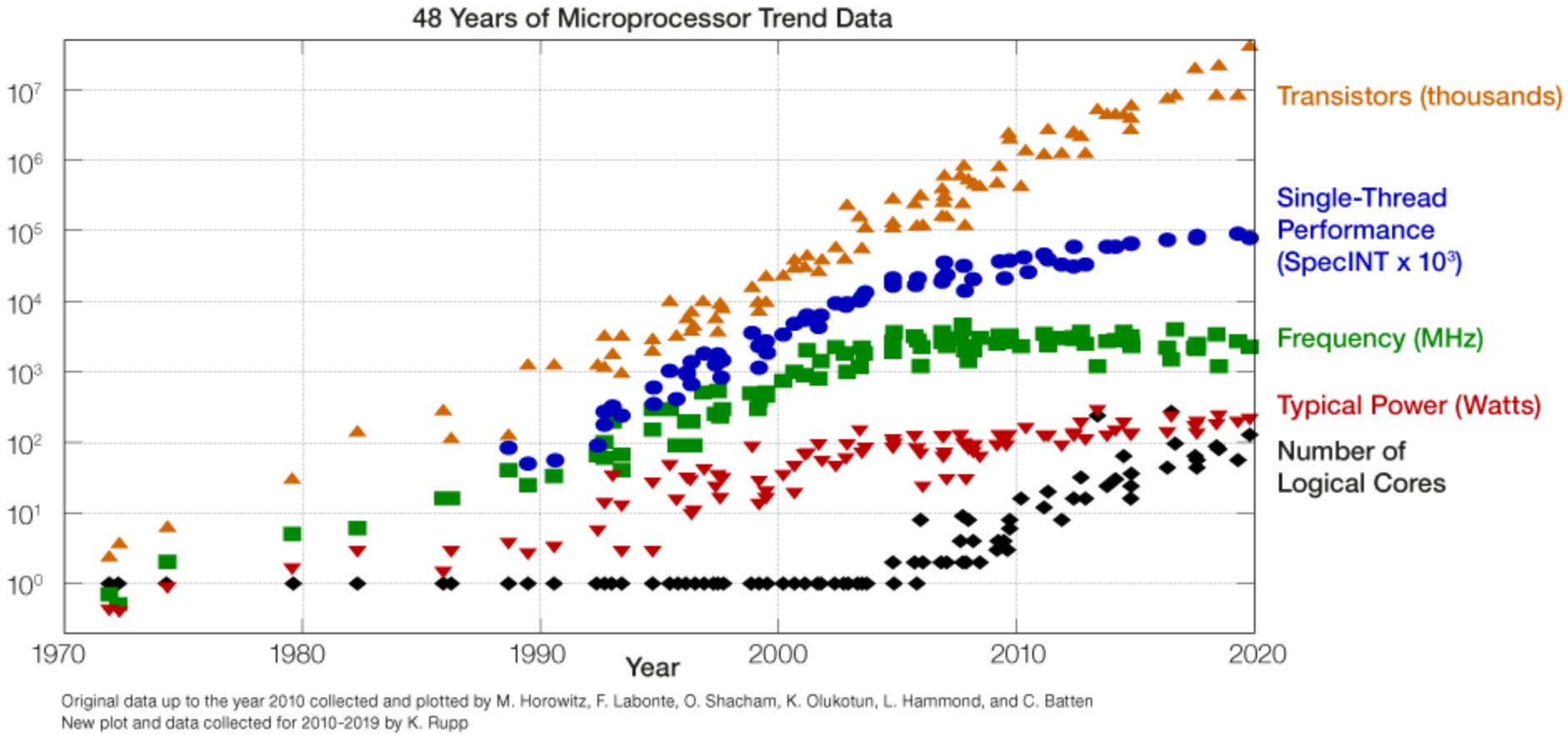
Moore's Law: The number of transistors on microchips doubles every two years

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

Our World in Data



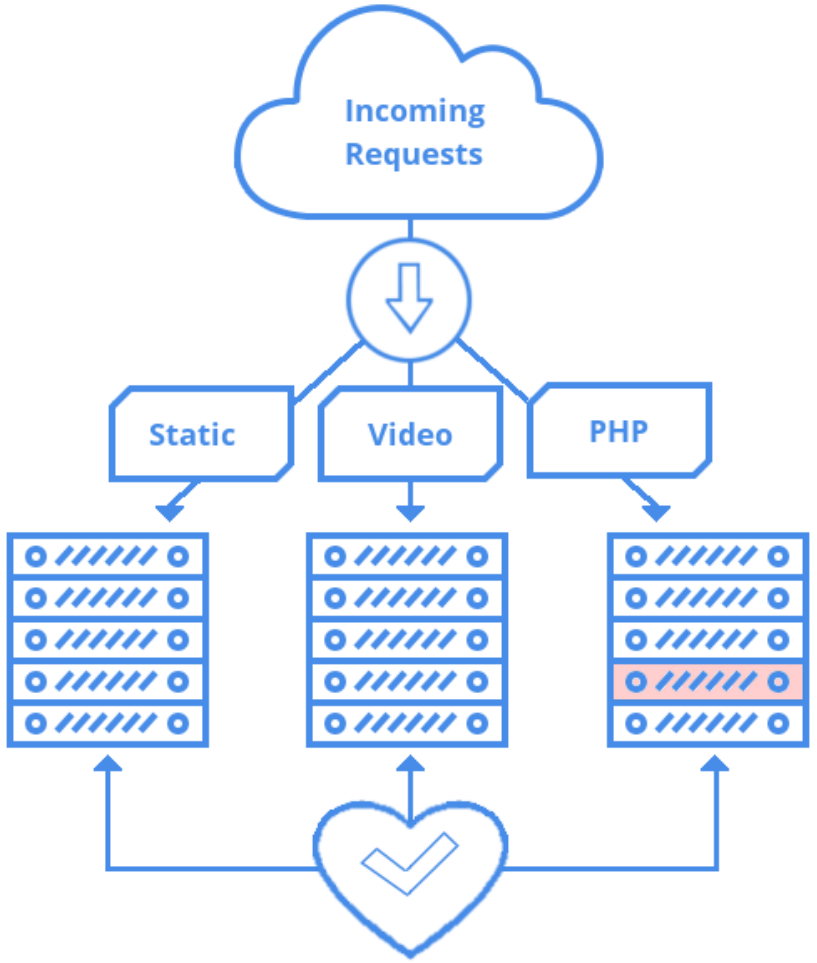
Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
OurWorldinData.org – Research and data to make progress against the world's largest problems.


48 Years of Microprocessor Trend Data
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
1970
1980
1990
2000
2010
2020
Year
Transistors (thousands)
Single-Thread Performance (SpecINT x 10^3)
Frequency (MHz)
Typical Power (Watts)
Number of Logical Cores
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp

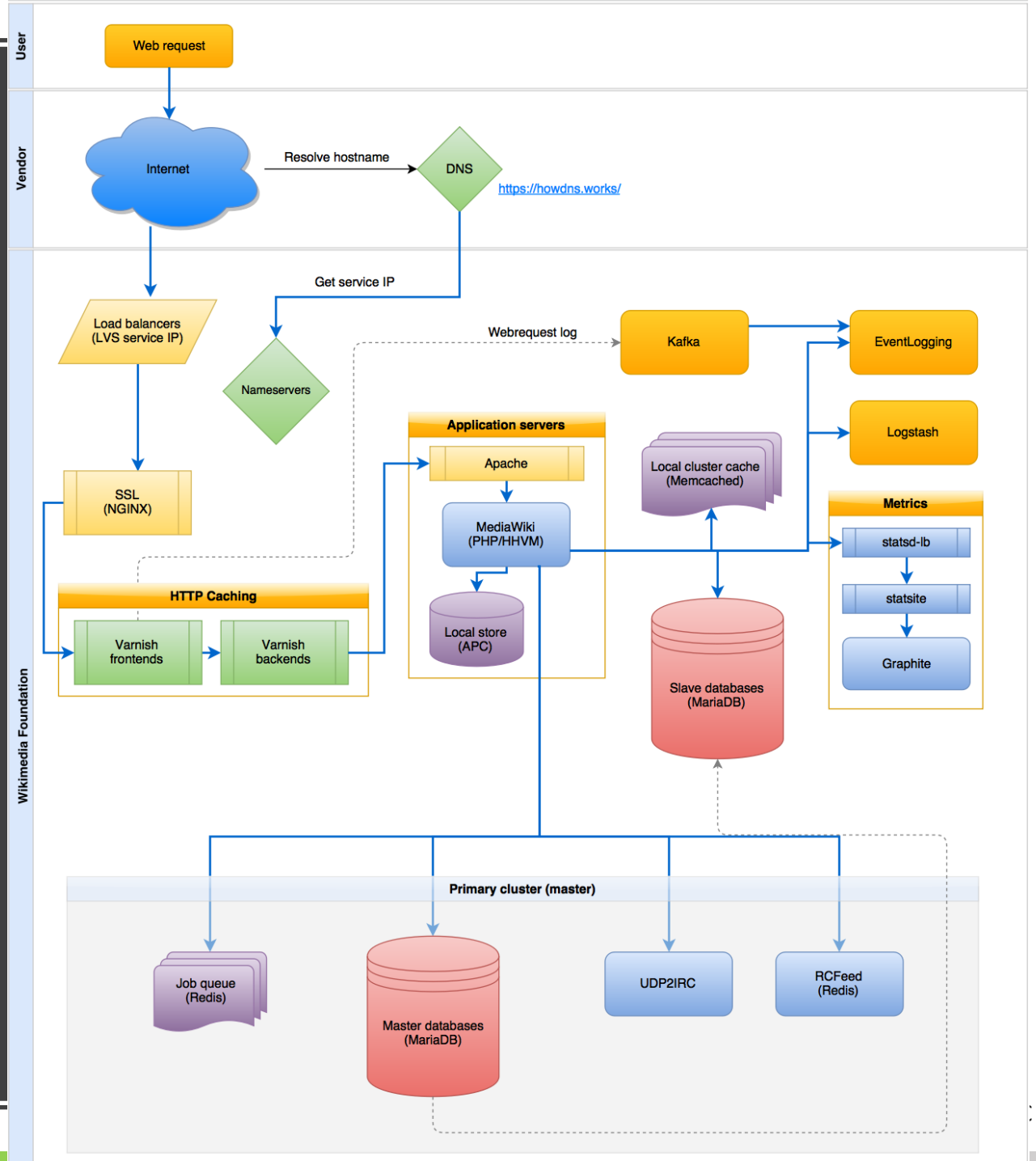
Why to Load-balance?

- **Facility scaling of 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



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)

Session 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 may be necessary in applications where 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
  - Ex: Redis, Memcached, Mysql, MariaDB

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)
- **Resilient Crossover will result in no error**
  - Most frequent in stateless applications

Load-Balancing Algorithms

- **Specific load based algorithms**
  - Look at or try to predict server load in determining the load of the real server.
  - Developed for each use case

- **least connections**
  - server with fewest number of flows gets the new flow request.

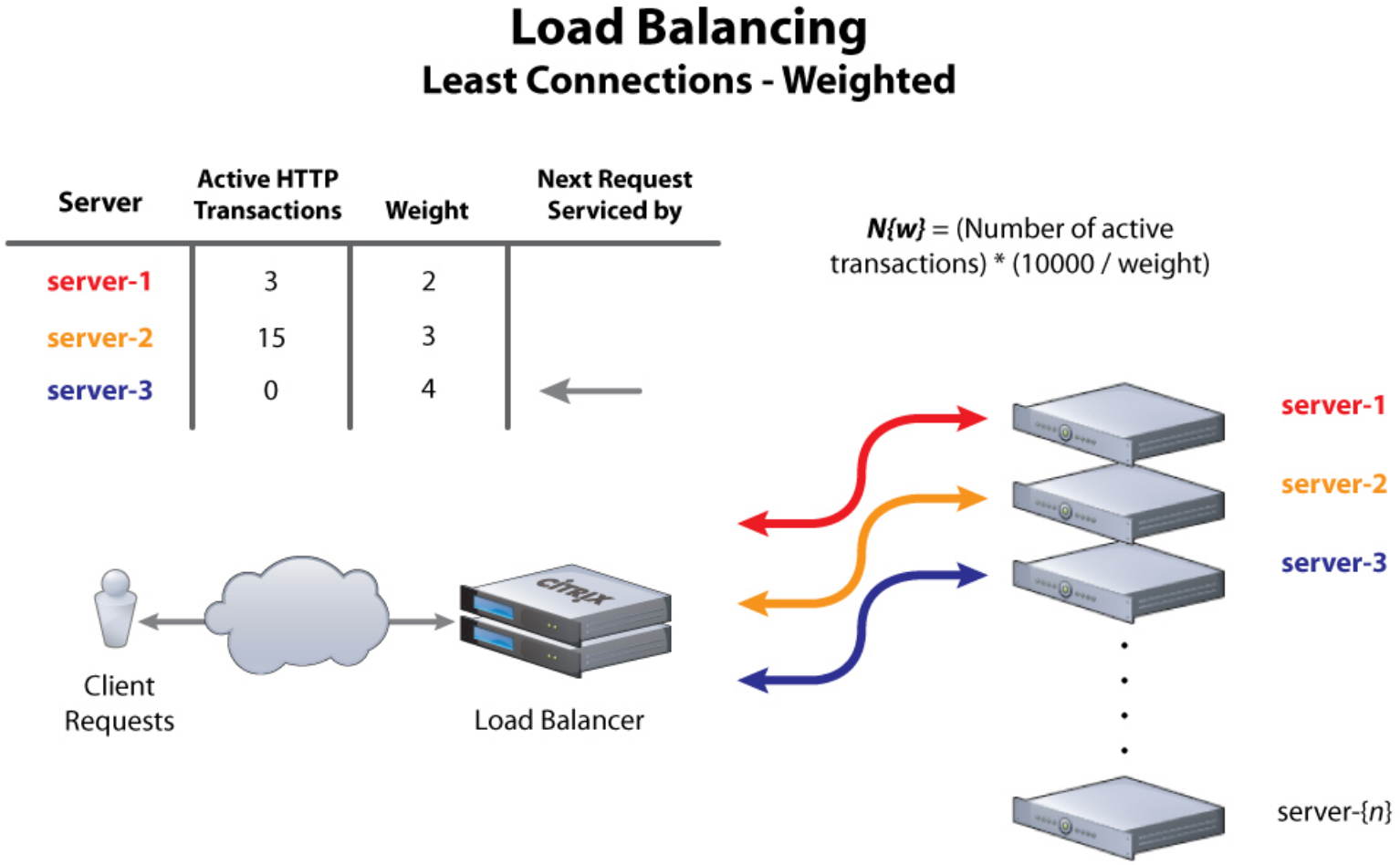
- **weighted least connections**
  - associate a weight / strength for each server and distribute load across server farm based on the weights of all servers in the farm.

Load-Balancing Algorithms

- **round robin**
  - Sequential attribution of requests to the servers in the cluster
  - most common algorithm
  - cheap and will distribute load among all servers equally
  - Problem: heterogeneous servers may be over/under utilization

- **weighted round robin**
  - give each server N number of flows in a row (based on a weight)
  - weight is set just like it is in weighted least flows.

Load-Balancing Algorithms



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
  - Involves the use of names, and not IP addresses

- **Local**
  - address returned points to address of instance, mostly in a round robin approach
  - Inside datacenter or cluster (e.g, kubernetes service)

- **Global**
  - address returned points to closest entry point, mostly for reducing latency
  - Closest cluster or datacenter
    - E.g: datacenter in the nearest availability zone

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.**

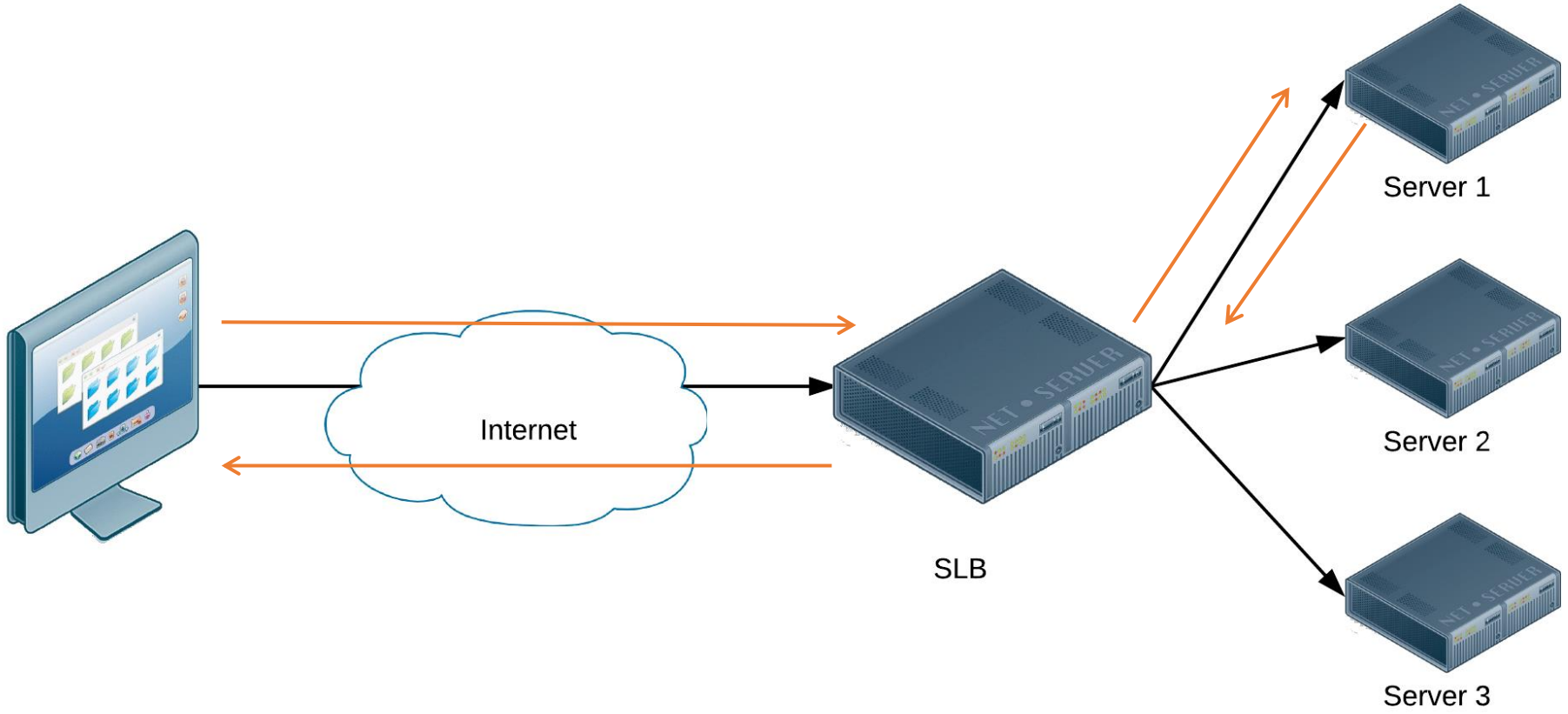
- **For some protocols, like FTP, you have to have knowledge of protocol to correctly load-balance (i.e., control and data connection must go to same physical server).**

IP - SLB Operation

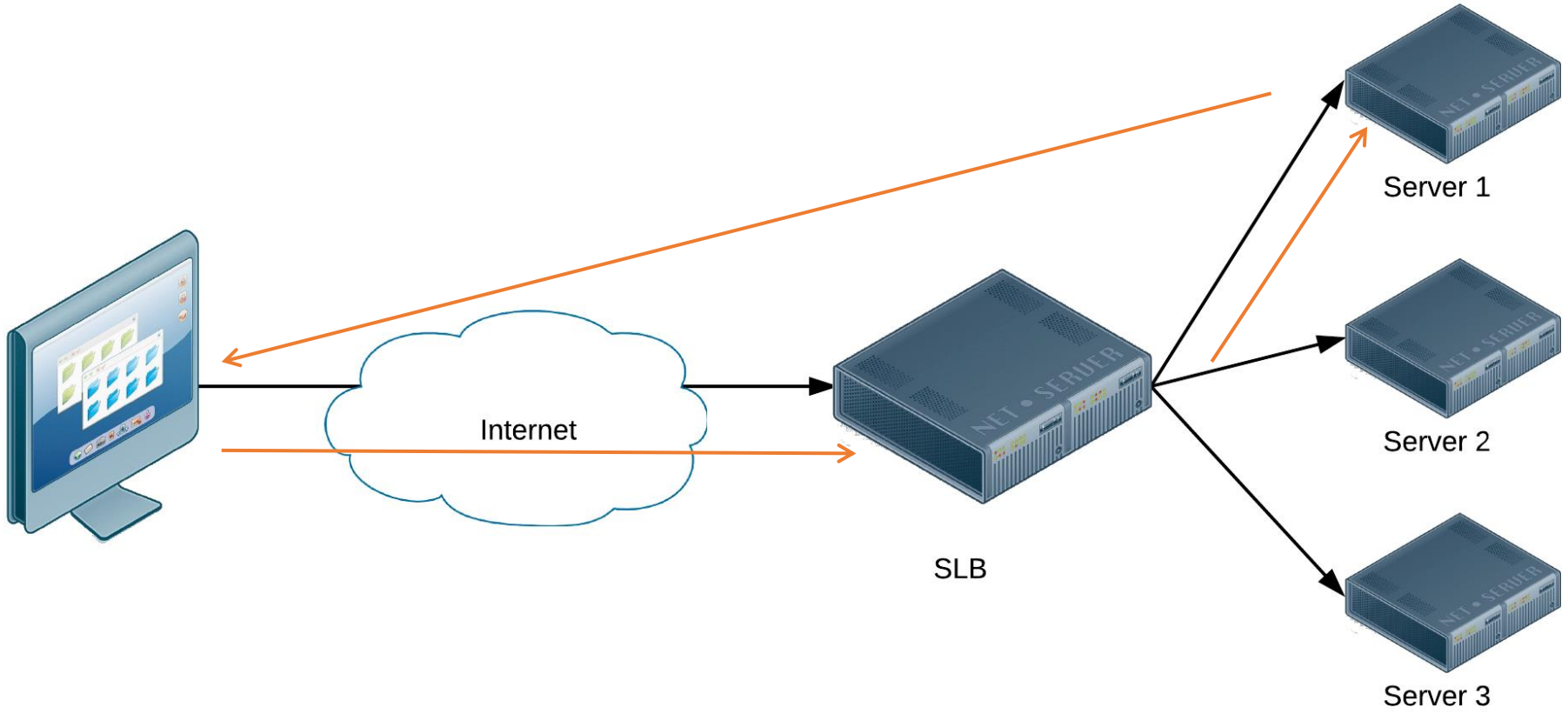
- **On a new flow, it determines if the virtual server exists.**
  - If so, make sure virtual server has available resources.
  - If so, then determine level of service needed by that client to that virtual server.
    - If virtual machine is configured with particular type of protocol support of session 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



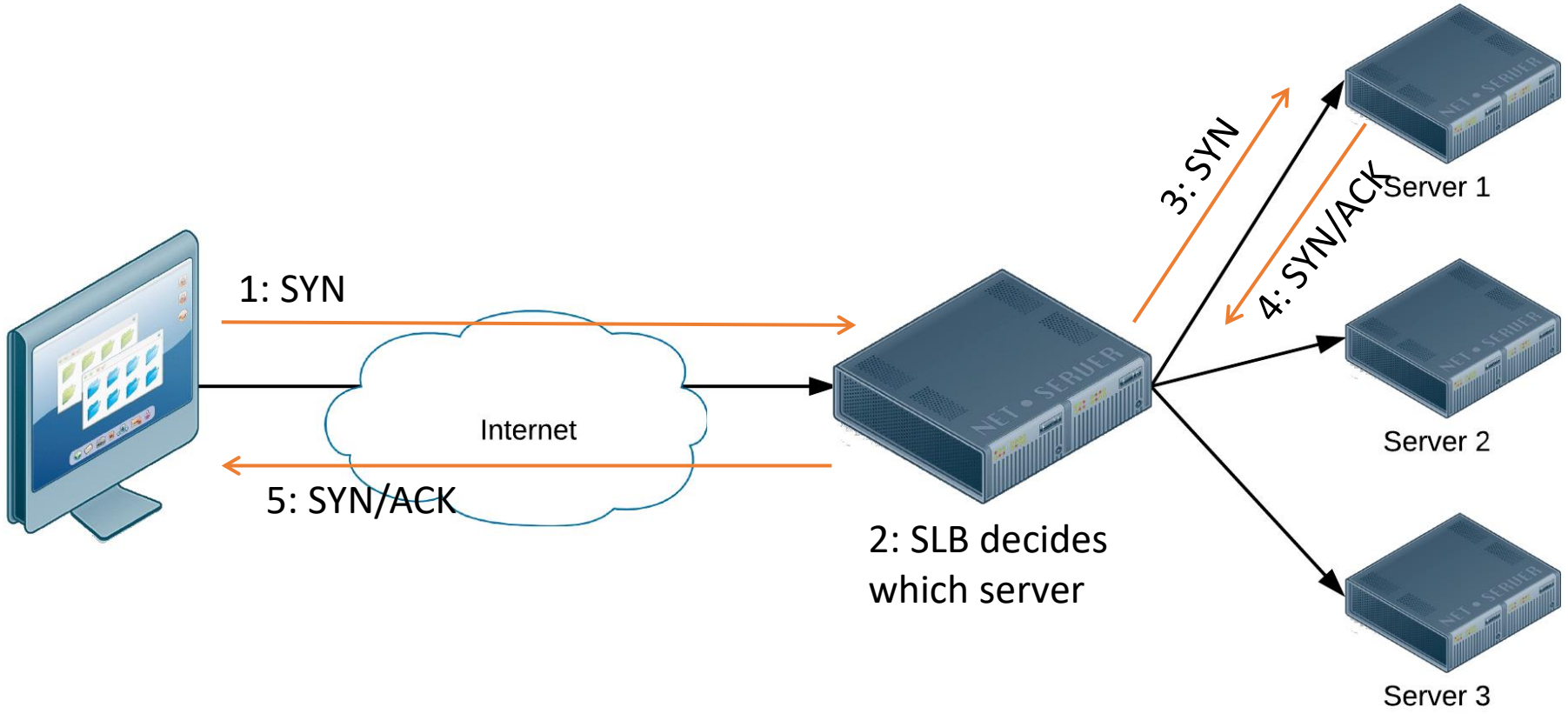
TCP - Load-Balance

- **Looking at the IP address and port to make a load-balancing decision**
  - Source and/or destination addresses and port
  - Destination address and port identify service
  - Source address and port identify client and origin

- **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

Equal Cost Multi Path

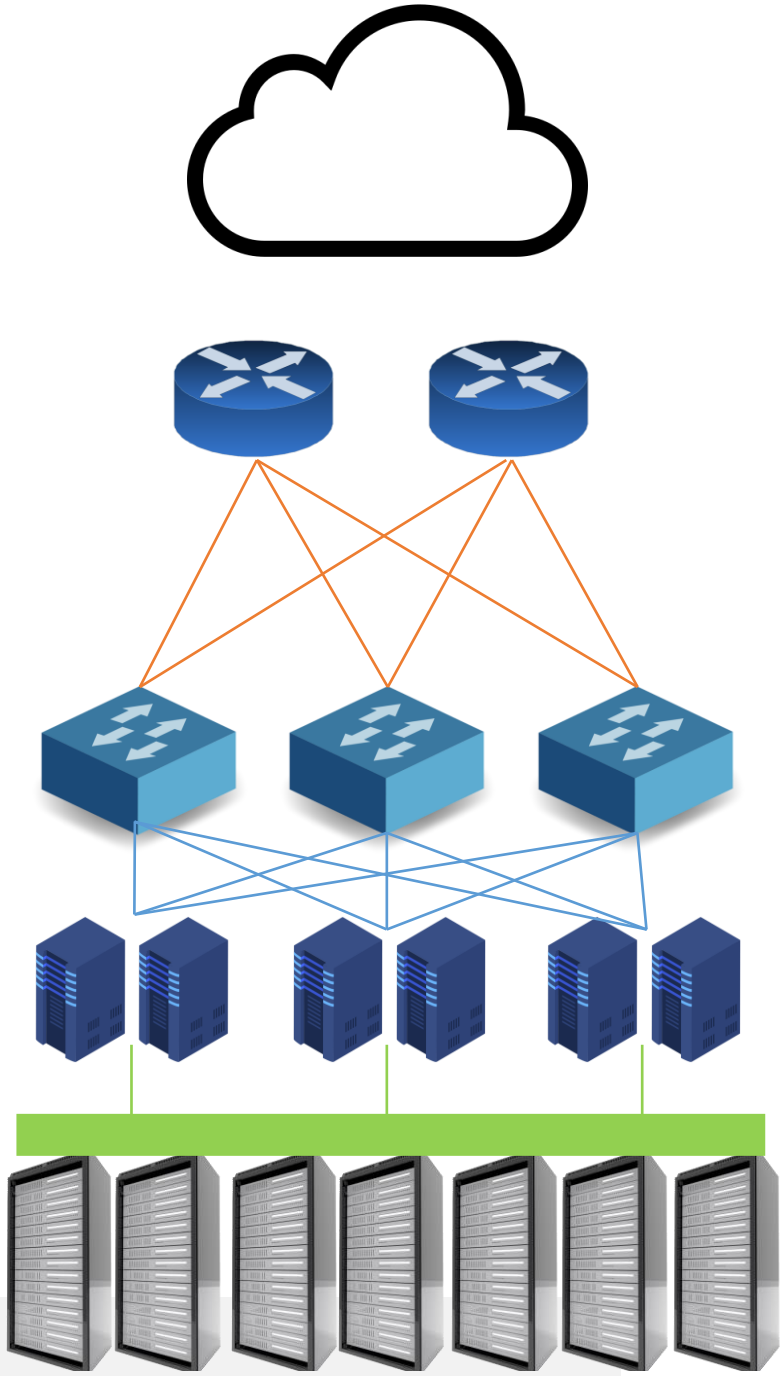
- **Considers that the entry router may have multiple routes to the same IP address**
  - These IP addresses may be in different servers!
- **Sends packets through different routes**
  - In the basic form, it's a problem for stateful traffic (TCP)

- **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

Equal Cost Multi Path

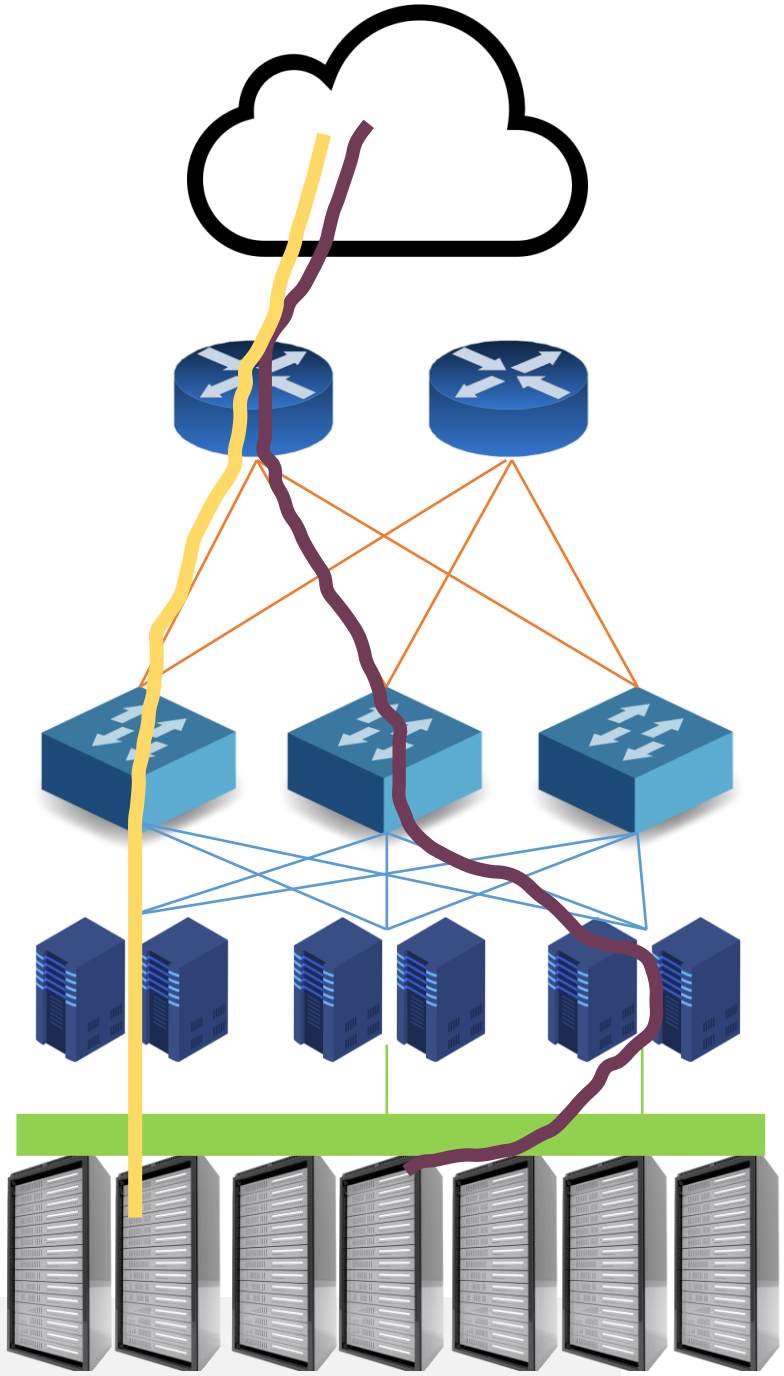
- VLANs, and other segmenting is possible
- Router typically dispatchs flow to higher layer LB
- Problems:
  - Hash colision
  - Link saturation
    - esp. w/ assymetric links



TCP - ECMP

Equal Cost Multi Path

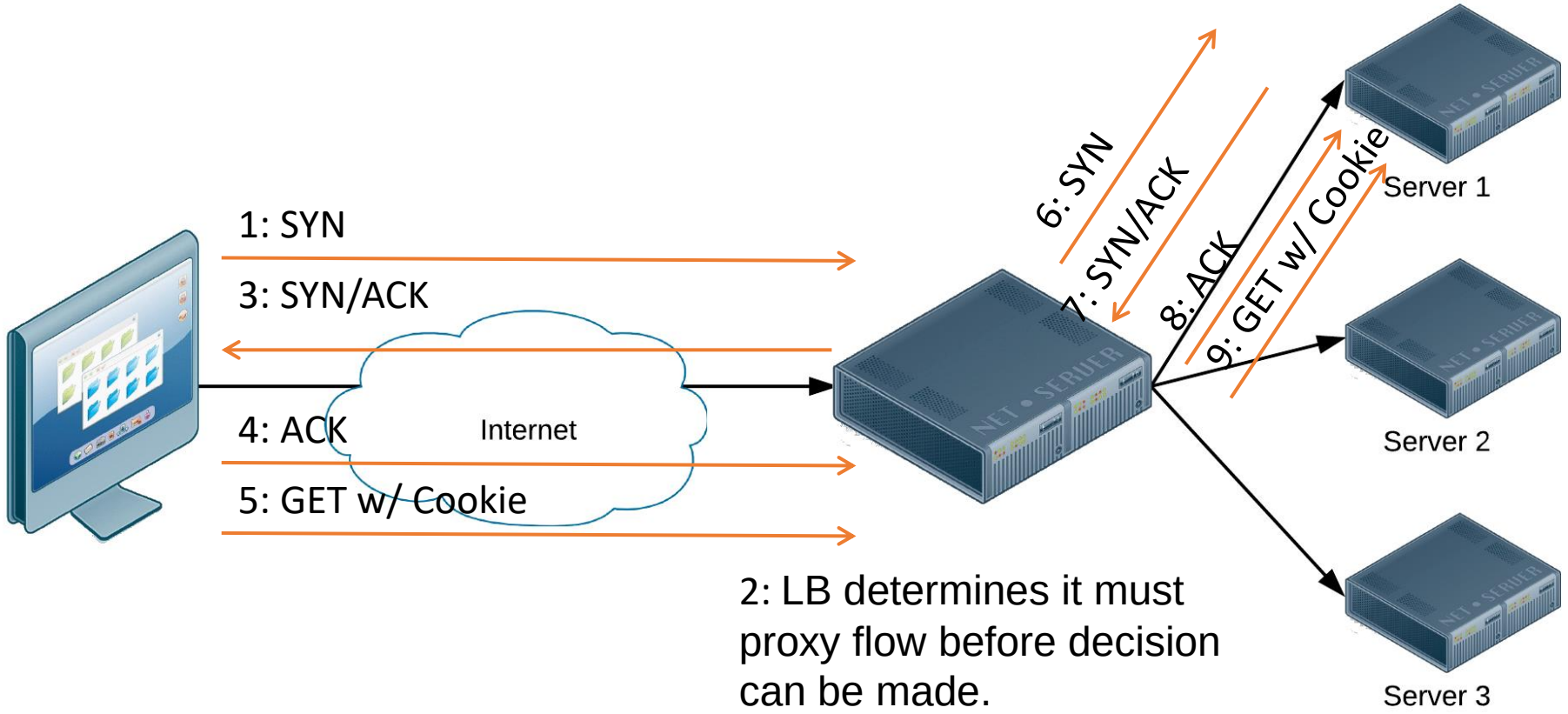
- VLANs, and other segmenting is possible
- Router typically dispatchs flow to higher layer LB
- Problems:
  - Hash colision
  - Link saturation
    - esp. w/ assymetric links



Layer 5+: Load-Balance

- **Load balance based on higher layer protocols and session state**
  - HTTP session and resource request
- **The LB must terminate the TCP flow destination 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 cluster**
  - 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.**

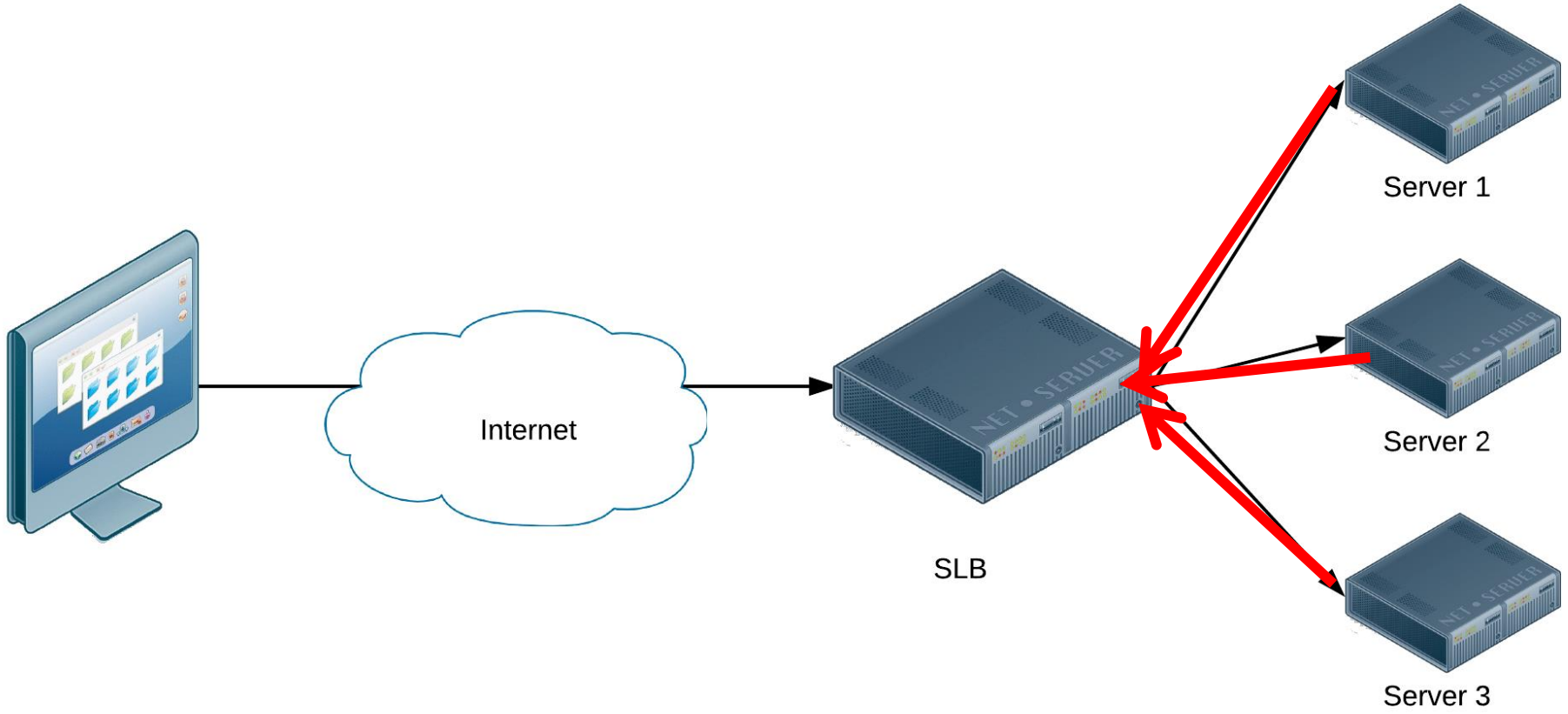
- **Start with statically configured value on LB device for initial start-up, then get weight from:**
  - Real server
  - Third party box / Collection Point
    - It is assumed that if a third party box is being used, it would be used for all the real servers in a server farm.

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

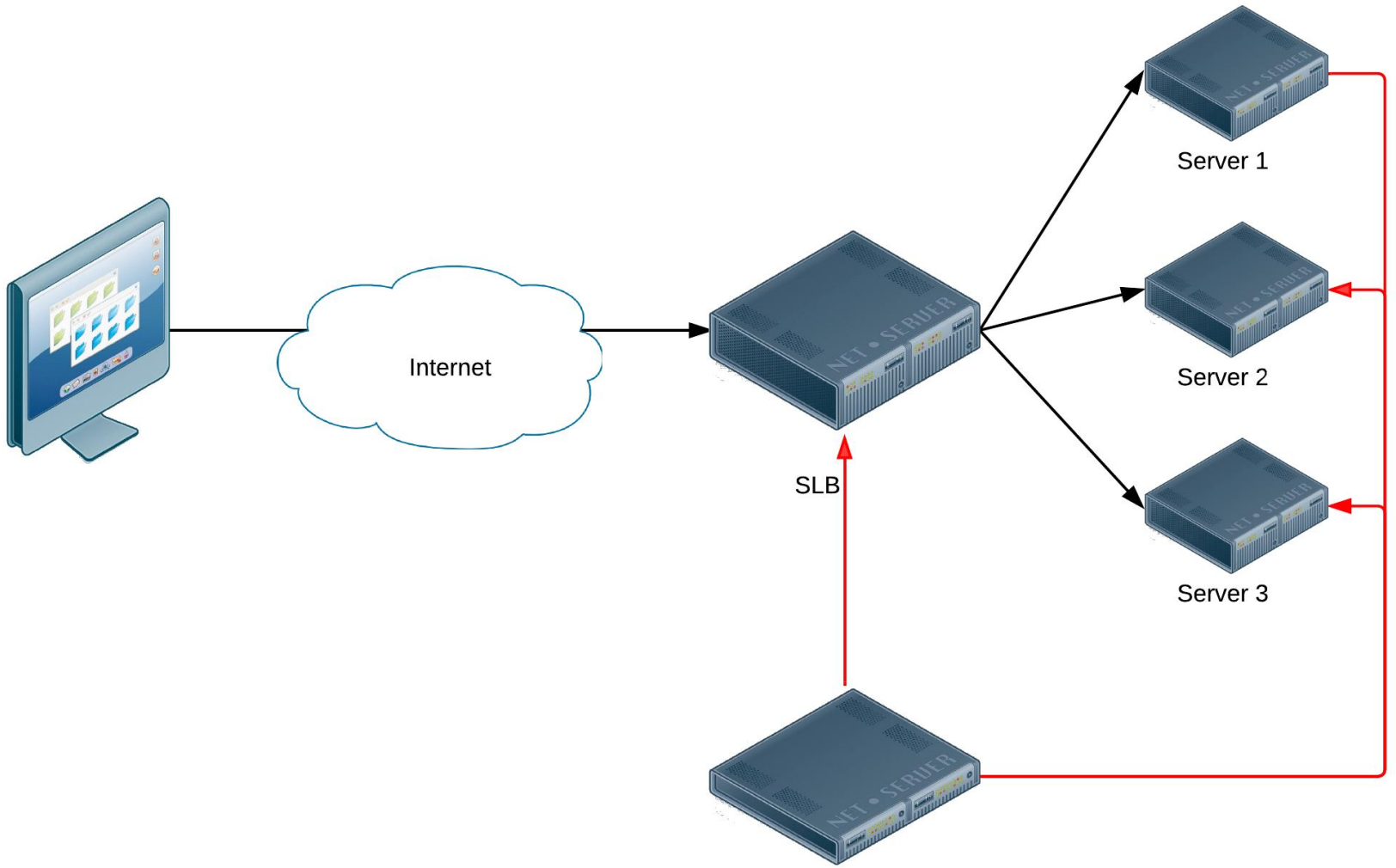
Pros

- 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, Prometheus, 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.