High Availability & Load-balance

System and Network Administration

Revision 2 (2020/21)

Pierre-Philipp Braun <pbraun@nethence.com>

Table of contents

What is Availability?

remember the CIA triad?…

==>

- Confidentiality
- Integrity
- Availability
- (Non-repudiation vs. Authentication)

Confidentiality

- symmetric ciphers

Integrity

- hash functions

Availability

- Fault-tolerance & SLA 99.999 (no downtime)
- DoS resistant
- High-load capable (scalable)

High-profile technical consultant

- Some guys look busy… (headphones seem to be useful for once)
- On-site at 3AM coordinating with operations
- Not just a sysadmin - high pressure and responsability
- Here comes the time when you get physical terminals
- …and no internet connection

The hidden networks?

- Public network
- Front-line facing nodes, gateways & Perimeter firewall
- DMZ / Perimeter network / CMZ
- Internal networks (user, voip, mgmt/backup)

What else exists?… (hint: non-routed networks)

==> Cluster networks (heartbeat & messaging)

==> Storage networks (SAN)

Types of clusters

- High Availability (HA)
- Load-balancing
- High Performance Computing (HPC) *aka* Grid

New categories in da place?

- Distributed storage
- Virtualization farm

High Availability Principles

- Know the basics before considering Distributed Systems
- HA used for decades by the industry for critical use-cases
  - Stock-exchanges
  - Space programs - we are talking about servers on earth and not Byzantine tolerant sattelite hardware
  - Telco (Telecom Companies)
- Subtile hence interesting acceptance testing
- Still powerful today e.g. KISS storage cluster
- Still needed anyway on the front-facing gateways and load-balancers

HA cluster software

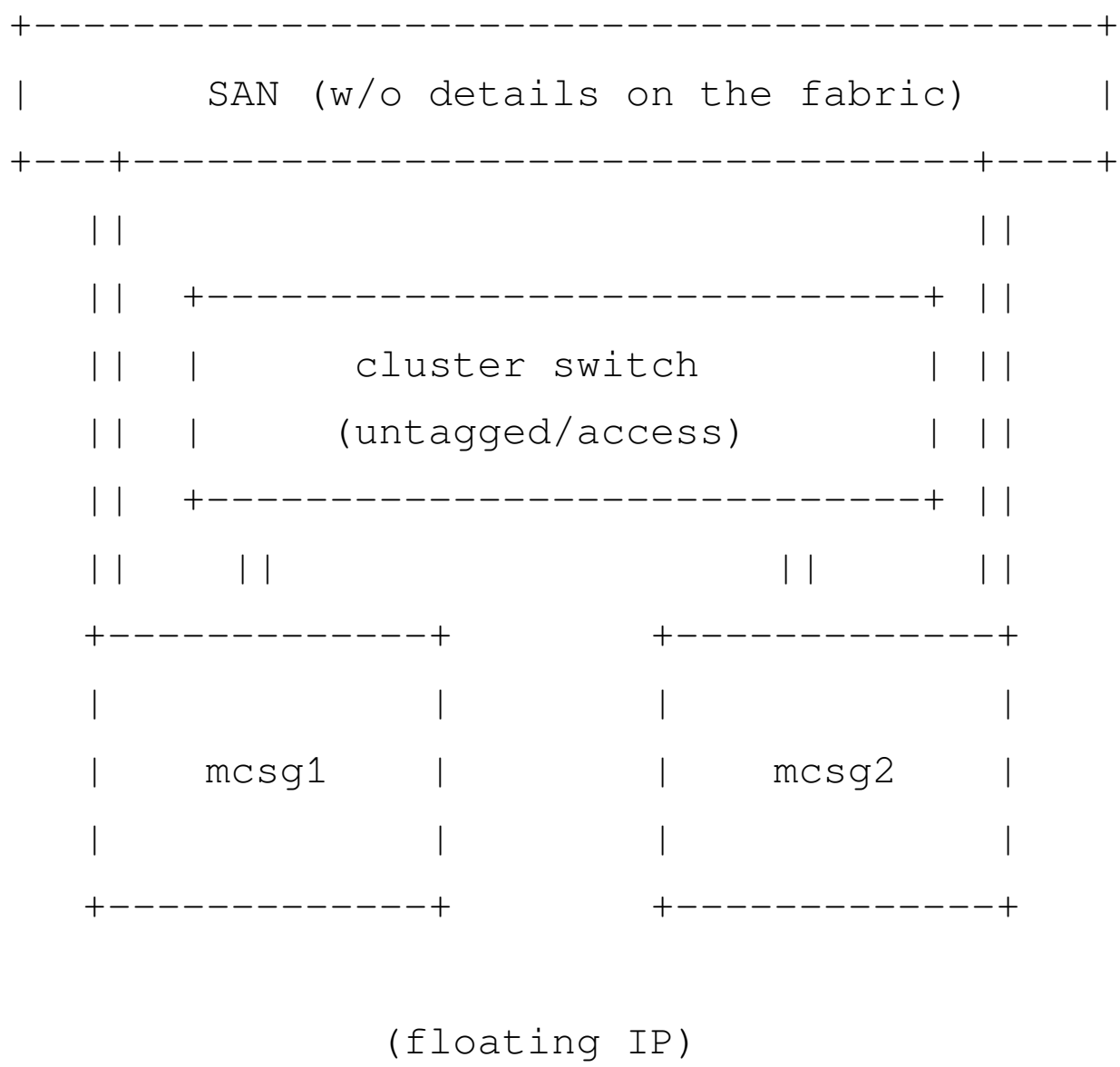
- HP MC/SG (MC/ServiceGuard)
- RHCS (Red Hat Cluster Suite)
- LinuxHA split
  - Heartbeat vs Corosync (heartbeat per-say)
  - Clusterlabs Pacemaker (messaging sub-system)
  - `crmsh` (SuSE, Ubuntu) vs `pcs` (Redhat)
- KISS `heartbeat` (serial, UDP, PPP/UDP)
- Veritas Cluster Server
- SteelEye LifeKeeper (now SIOS Protection Suite for Linux)
- IBM PowerHA SystemMirror (formerly HACMP)

MC/SG example use-cases

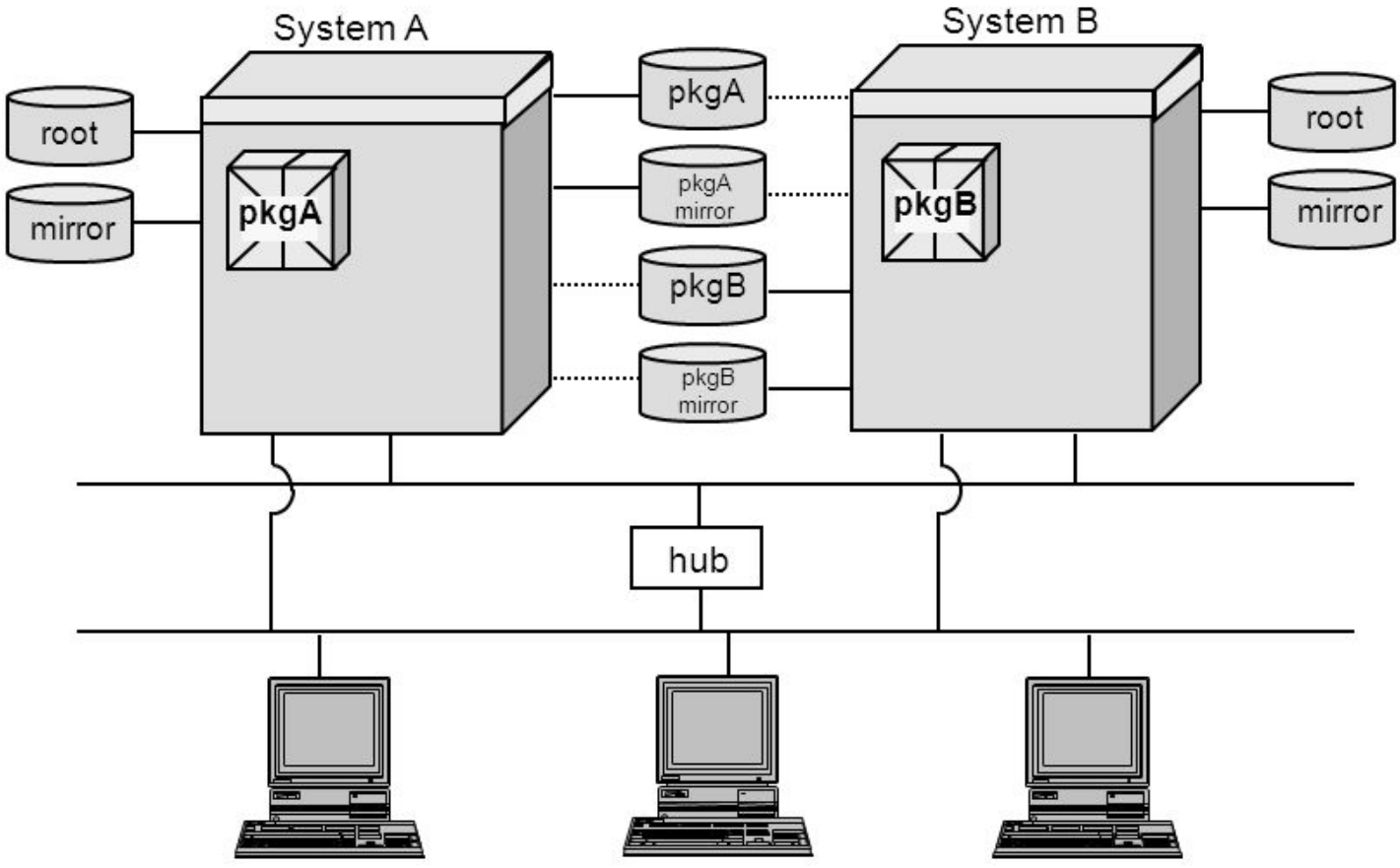
- MC/SG with Oracle Database at Tokyo Stock Exchange
- MC/SG with proto-cloud contacts at SFR Telco
- you may want to automate things for system preparation
  - network setup
  - package requirements
  - …

hence some distributed shell for 3+ nodes

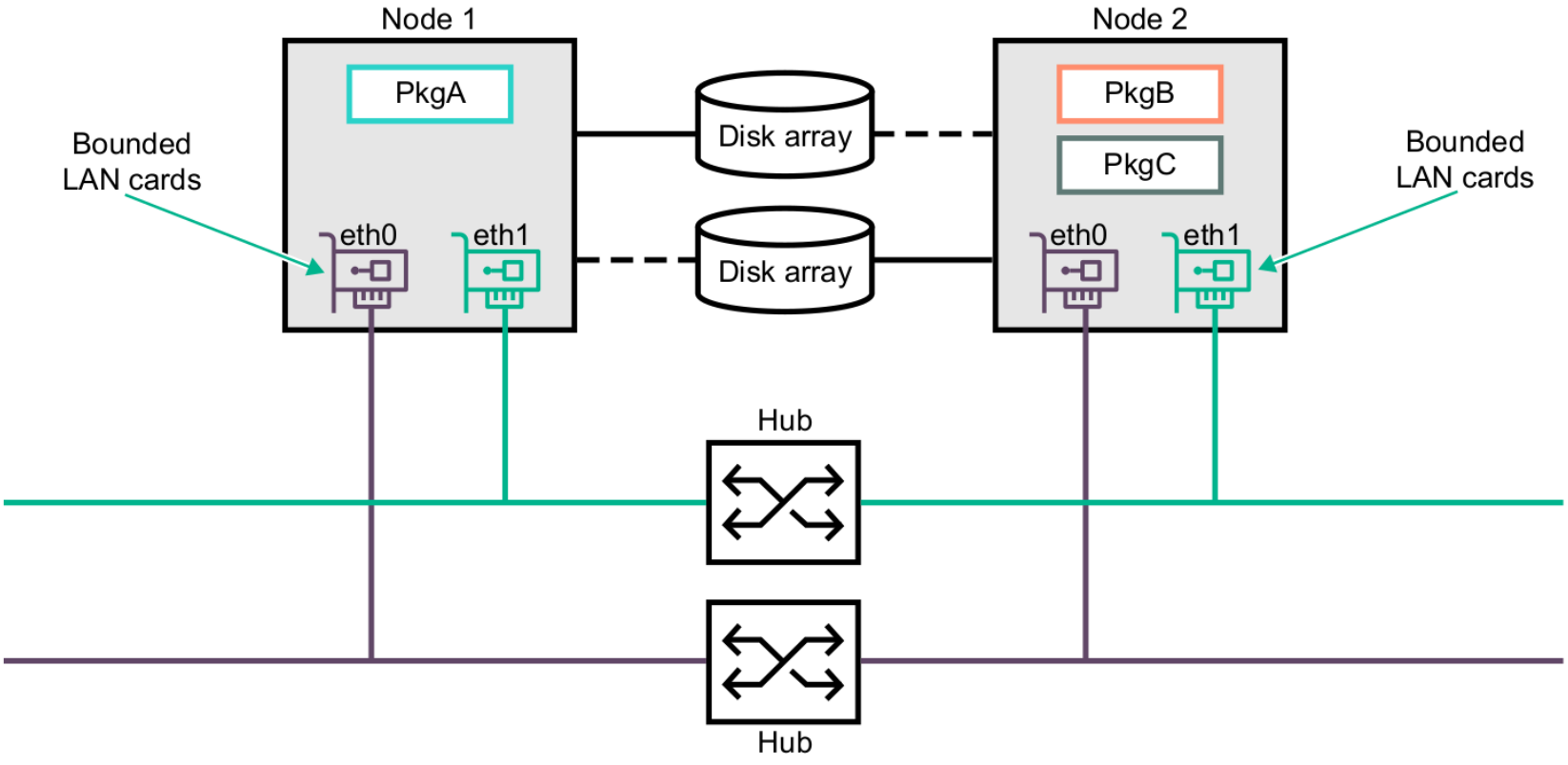
This is tradition



Introducing MC/ServiceGuard



Old school active/active



New graphics for old school

HA resources & packages (resource sets)

- Need to associate set of resources as *Packages*,
- e.g. an `NGINX` service and the `Floating IP` that goes with it

HA active/active != network load-balance

- There may be multiple *packages* (and `floating IPs`)
- Usually full active/passive (all *packages* on one node only)
- Active/active meant balanced active/passive *packages*
- This is not for cluster-ready and non interfering Docker instances
- But rather for old-school critical and usually monolitic services

How to HA?

- Fault-tolerance at every level

What for?

- reduced down and maintenance time
- SLA 99.99 –> SLA 99.999

Fault-tolerance at every level

no SPOF anywhere

- Datacenter 2N architecture
- Datacenter cooling
- Enterprise-class server & RAID controllers
- Cluster software

2N architecture

- Data-center's energy supply redundancy
- Meaning fully redundant and differentiated supply sources,
- with at least one having power-generators as backup

Free cooling

Free-cooling (Facebook in Sweden)

Kill animals

2nd pass air filters

Rackmounts in a bath

Enterprise-class servers

- HPE ProLiant & BladeSystem
- Dell PowerEdge
- IBM xSeries
- SuperMicro
- Fujitsu
- Huawei (esp. TaiShan…)

DL585

Two disks are enough - RAID-1 for the system to boot

DL585 rear - two PSUs

Recap: enterprise-class servers

- Xeon vs Opteron
- Lights-Out Management (LOM), always
- *Or go for ARM (Chinese are comming…)*

Hardware RAID controllers

- HP/Compaq SmartArray (`cciss_vol_status`)
- Dell - PERC (they have their own interpretation of RAID-1)
- IBM - LSI/Broadcom Megaraid (WebUI is ugly as hell)

Software RAID products

- mdadm (GNU/Linux)
- LVM2
- ZFS
- RAIDframe (NetBSD)
- …

Low-cost & DIY

without enterprise-class servers

- keep having Opterons or ARM
- live without certification matrixen
- you get the change to secure your firmware (Coreboot & friends)
- consume a less energy (80 Plus)
- remember you cannot manage what you cannot measure - and you need tools for that
  - monitor temperature
  - get alerts for unexpected excess consumption
  - analyze & diagnose waste (and forget about Bitcoin mining, it's too late anyhow, right?)
- …and buy a bunker (or a Faraday cage)

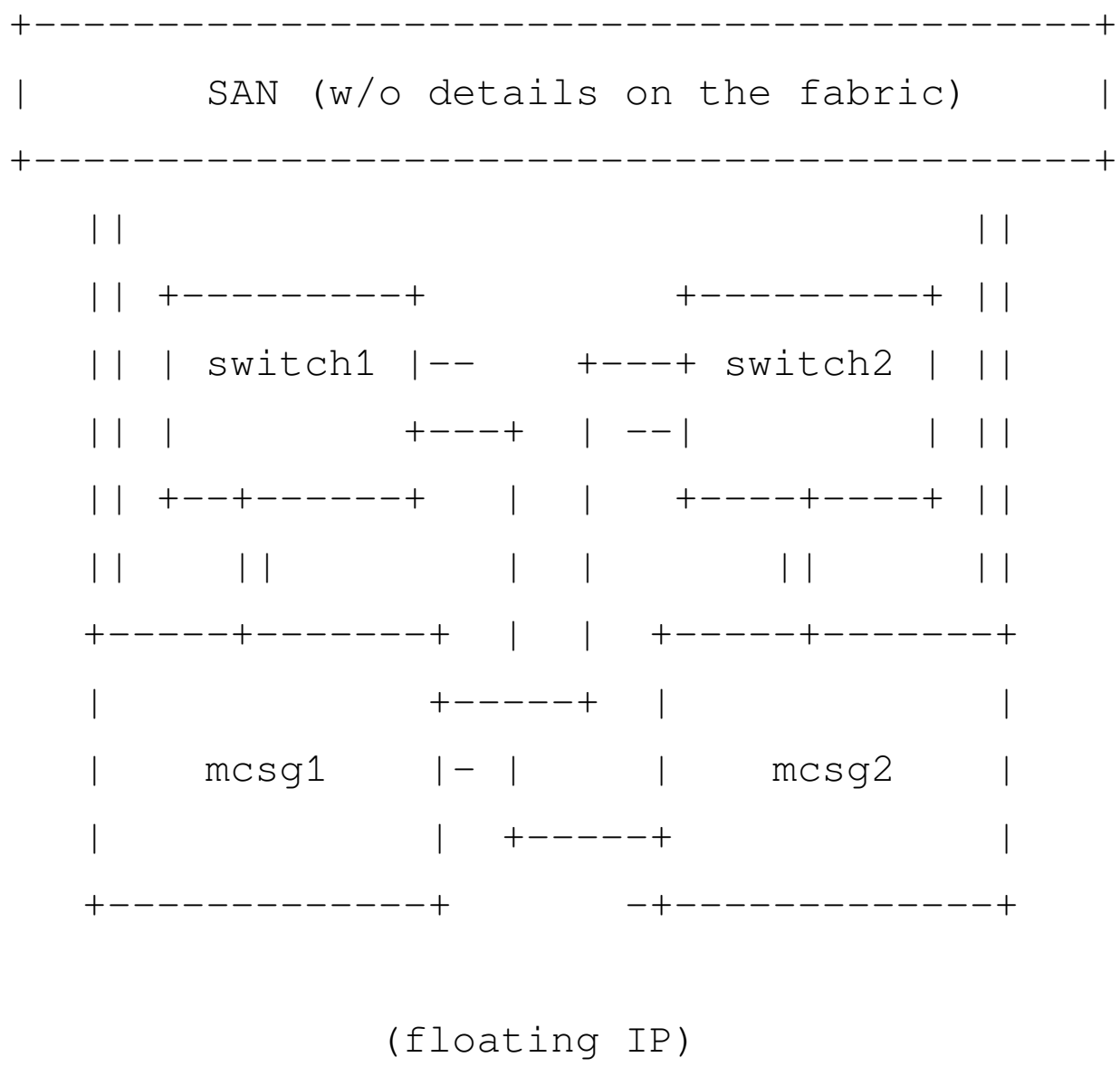
Cluster design

Anything wrong with the ASCII diagram from above?…

(remember every component should be doubled)

REDUNDANCY EVERYWHERE

(TWO CLUSTER SWITCHES)



- But then we need to reduce the LACP pipe! (needs to be done on the switch also)
- *(Unless another link aggregation method copes with that)*
- We're loosing money! (not a problem - actually it is the contrary as we are selling more IT service)

SLA 99.999 thanks to active/passive trickery

No time for downtime…

- Hardware upgrades
- Firmware upgrade
- System upgrade
- HA software upgrade
- ~Application upgrade
- –> Could also do just restart after local upgrade,
- –> however the replicated setup makes it an easy rollback

// Any questions on the principles of HA?

Fault-tolerance

https://xkcd.com/705/

Old-school HA

The simpliest distributed system ever

```
active nodes > total nodes / 2
```

- Use of **quorum + fencing** to avoid split-brain situations
- Ideally an odd number of nodes
- MC/SG Lock LUN - dedicated small LUN where memberships get written
- MC/SG Quorum server - PING & LOCKS (DEDICATED SERVER JUST FOR THAT)

Fencing / STONITH

- A node is failing? Shoot it down (or reboot it)
- Using a layer below e.g. UPS/PDU and/or BMC (the cluster quorum has access to that)
- Non-cluster-ready app should not be started twice
- Prevent split-brain and data corruption

Floating IP

This is the one traditional cluster software is using

- Floating IP (resource-agent script is stairghtforward, possibly DIY with `ip`/`ifconfig`)
- MAC address differs
- IP get simply taken over by the other node

–or– VRRP vs. CARP

VRRP

- LAB // is there VRRP available on linux to play with?

CARP

- there is on NetBSD, OpenBSD, FreeBSD, …
- Linux has `ucarp`, though

// Questions so far?

High Availability Acceptance Testing

or why HA-specialized IT consultants do exist

Network redundancy

- “Pull each cable and check bonding”
- “Kill primary switch”
- “Kill secondary switch”

Cluster is still under control

Enterprise-class server takes time to boot so you will have time for multiple coffees

- “Controled migration & node withdrawal”
  - “Restart the node and rejoin the cluster”
  - “Repeat for the other node”
- Controlled out of cluster service simple power-off

Nodes are crashing

- “Crashing nodes ; the ultimate test”
  - “Ensuring crashed node is fenced”
  - “Confirm all services recover on the surviving node”
  - “Rejoining the recovered node and migrating services back”
  - “Crashing the other node, ensuring its services recover”

Fencing acceptance example 1

Emulate a system crash with Magic SysRq key (reboot)

```
echo c > /proc/sysrq-trigger
```

What to expect: fencing should reboot by a BMC call

Fencing acceptance example 2

Pull the power plug(s) on a server.

What to expect: fencing should shoot that node by any means

The fork-bomb test

Make sure services really crash and reboot instead of turning into a loop

- Just in case, try a fork-bomb on a node
- Tune the status scripts at a higher-level
- –> consider a service failing of that does not respond

// Questions on fault-tolerance and fencing?

Old-school was really painful, is there anything new and simpler?…

Btw did we solve the A from the CIA triad?…

==> Solved

- Fault-tolerance & SLA 99.999 (no downtime)

==> Not solved *yet*

- DoS resiliant
- High-load capable & scalable
- DDoS resiliant

Scalability

First, how to be DoS resilient?…

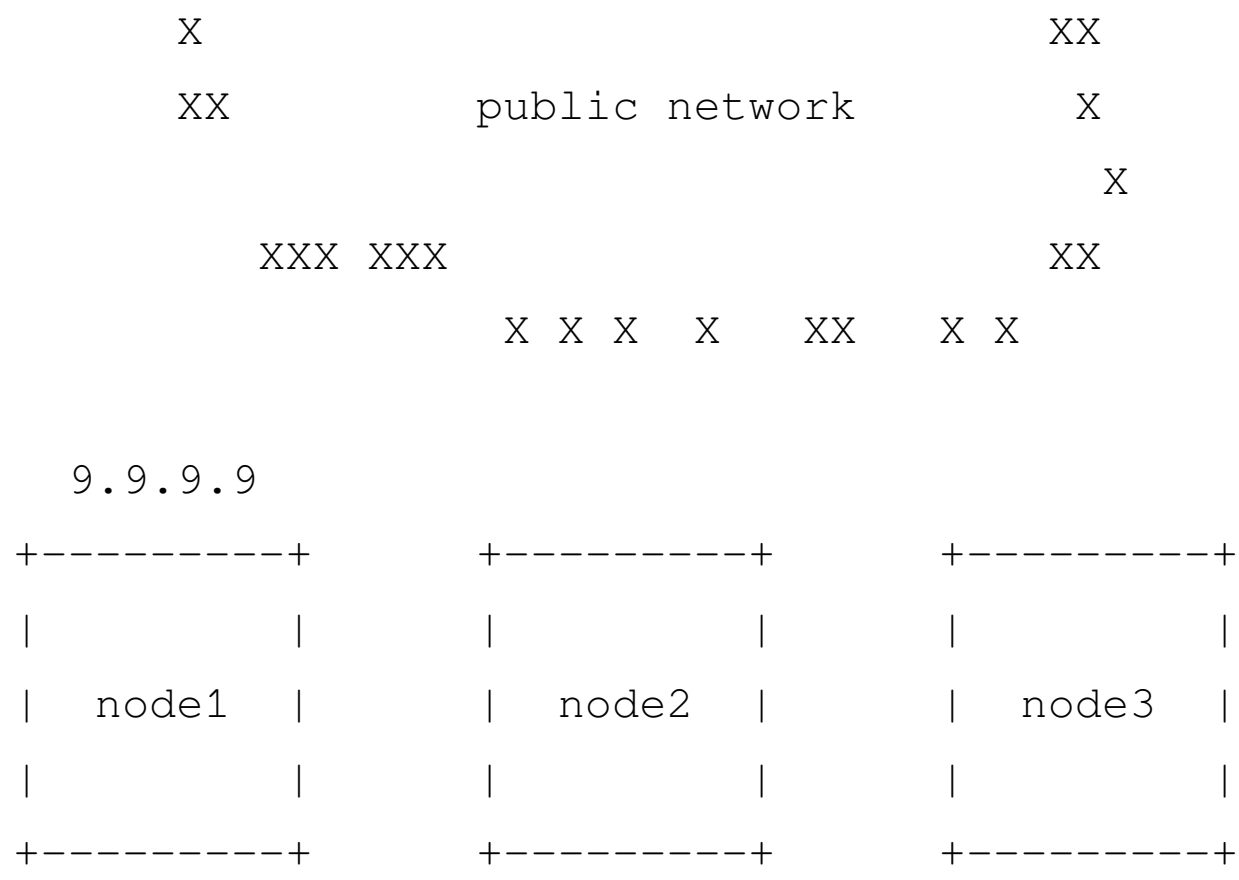
==> infrastructure *and* code done right

- Principle of least privilege
  - firewall policy
  - cluster services need to listen internally only
  - service/app authentication & authorization
- KISS / no hype required, less code == less vulns == less bugs
- Code harden & pay devs to REMOVE lines
- Fast incident response (about monitoring and well-established HA processes, not about forensics)

Second, how to be high-load resilient?…

==> load-balance ready

- Load-balance cluster – apps are stateless and/or cluster-aware
- Distributing the network load is the true active/active
- Shared data storage
- Big-enough pipes



- got Swarm or K8S cluster
- one application end-point with public IP `9.9.9.9`
- DNS record `app.example.net IN A 9.9.9.9`

Everything is fine there?

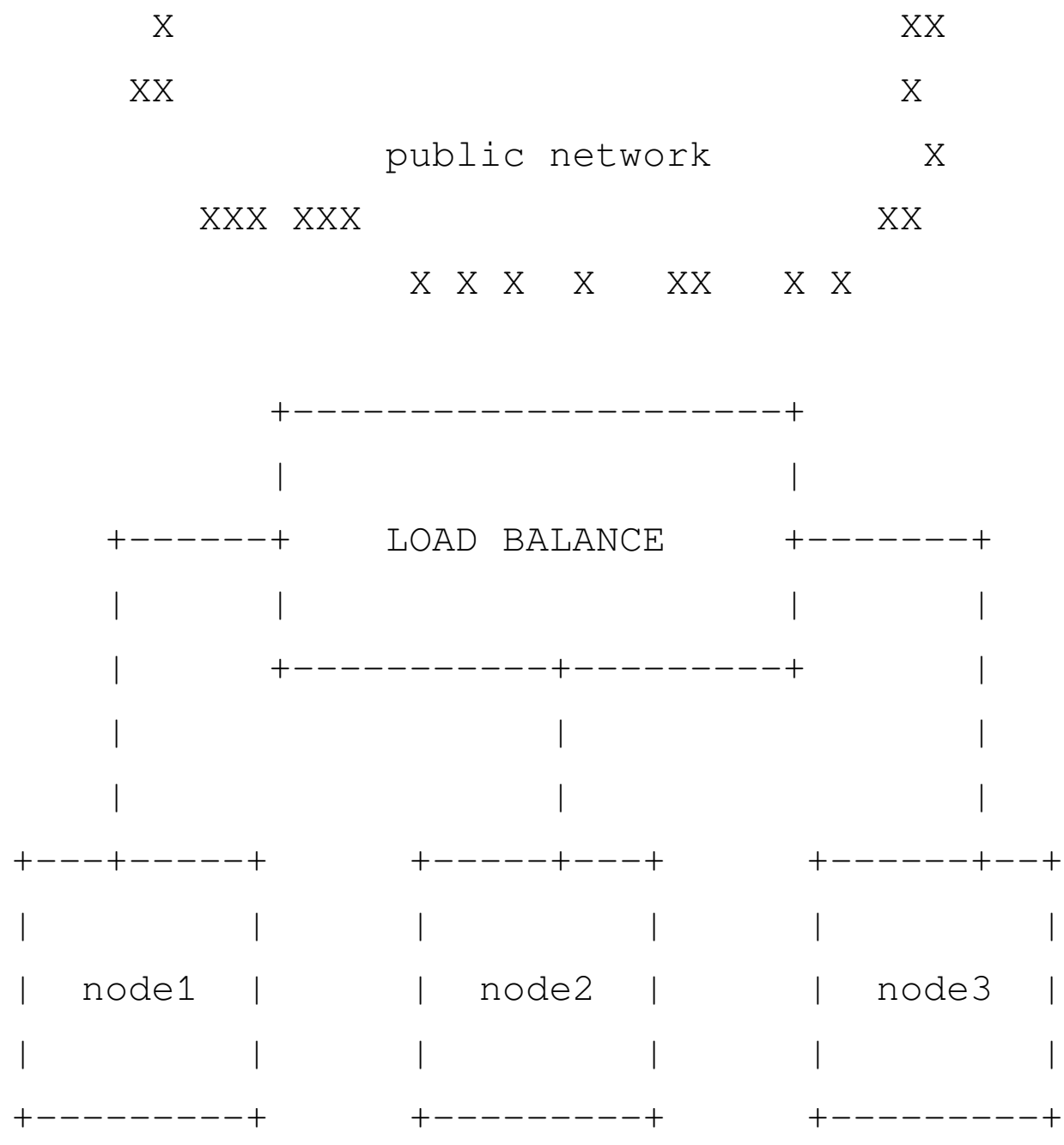
==> NO – all the load goes to only one node

- Swarm and K8S do have a network overlay by default, which re-distributes load to other nodes
- however node1 becomes a load-balancer *ipso-facto* here
- node1 is not necessarily sized for that purpose

How to design `app.example.net` *so the traffic gets shuffled around the nodes?…*

==> TWO SOLUTIONS

- either by means of DNS round-robin
- –or– by means of a load-balancer in between


public network
LOAD BALANCE
node1
node2
node3

Balancing methods

- Layer 3 round-robin
- Layer 3 round-robin **with “sticky connection”**
  - remind src/internal-dst IP addresses
- Layer 7

Commercial load-balancers

not sure what part is truly hardware-based

- F5 BigIP
- Fortinet FortiGate
- …

Open Source load-balancers

- layer-3 BSD `pf` vs. `npf` vs. `ipfilter`/`ipnat`
- layer-3 Linux Netfilter (`iptables` vs. `nft`)
- layer-3 eBPF // LAB PoC that!
- layer-3+7 HAProxy
- layer 7 NGINX / NGINX Plus (dynamic objects?)
- layer 7 Apache Traffic Server? (static objects?)
- layer 7 OpenBSD Relayd
- K8S Ingress / Ingress-NGINX

So can we just replace the previously seen HA setups with load-balancing?

May we simply forget the tradition?…

load-balancing != HA

==> Yes and No

- Yes as long as orchestrator manages the instances **and VIPs**
- No if nothing takes care of the cluster health already

HA-capable load-balance

Which ones are HA capable against the back-end nodes/services?

==> With additional scripts, probably any – though maybe not that corporate nor resilient

==> Built-in for sure *aka* **Health Checks**

checking the back-end service

- layer-3+7 HAProxy
- layer 7 NGINX / NGINX Plus
- layer 7 OpenBSD Relayd? // LAB PoC that!

think of fault-tolerance again

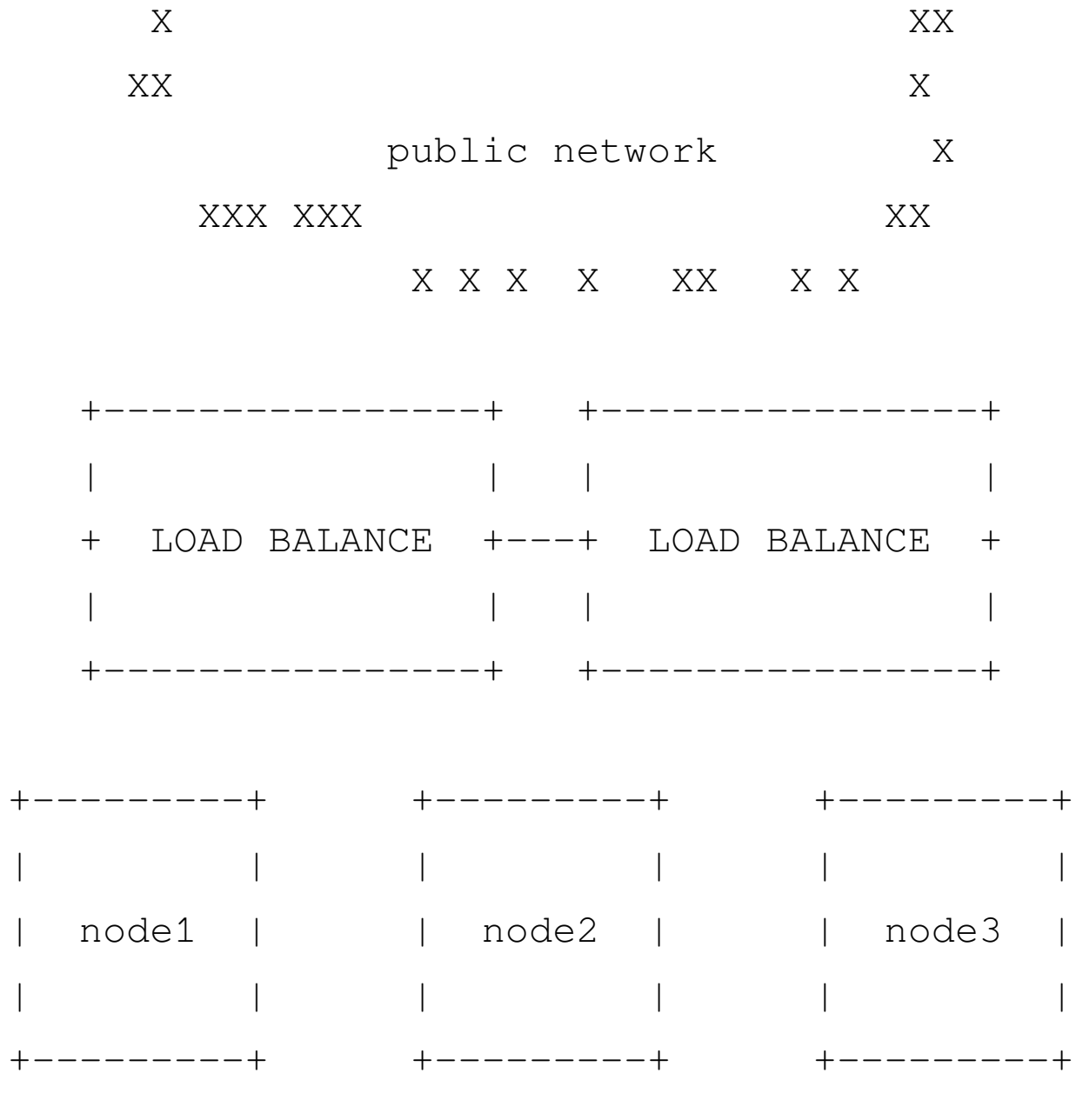
What was missing from the diagram above?…

Redundant load-balance

==> THERE WAS A SPOF!

The balancers need to be fault-tolerant also

ACTIVE/PASSIVE - NO DNS ROUND ROBIN



Which balancers can be made fault-tolerant themselves?…

==>

- `pf` with `pfsync`
- any other filter as long as you keep the configuration in sync somehow
- and enable some IP failover (VRRP/CARP is better than a VIP)

when it is more than just high load to handle

How to resist DDoS attacks?…

==> DDoS resilient

DDoS Protection at some NOC & ISP

- Arbor® Networks Peakflow
- Sevi® M6-NG

CDN - *a world-wide load-balance scenario*

- got more reverse-proxies than backends
- your backend is unknown to the end-users
- your DMZ is somehow the public network itself

BGP 666 black-hole at some NOC

- ISP only looses one customer

// Questions on scalability?