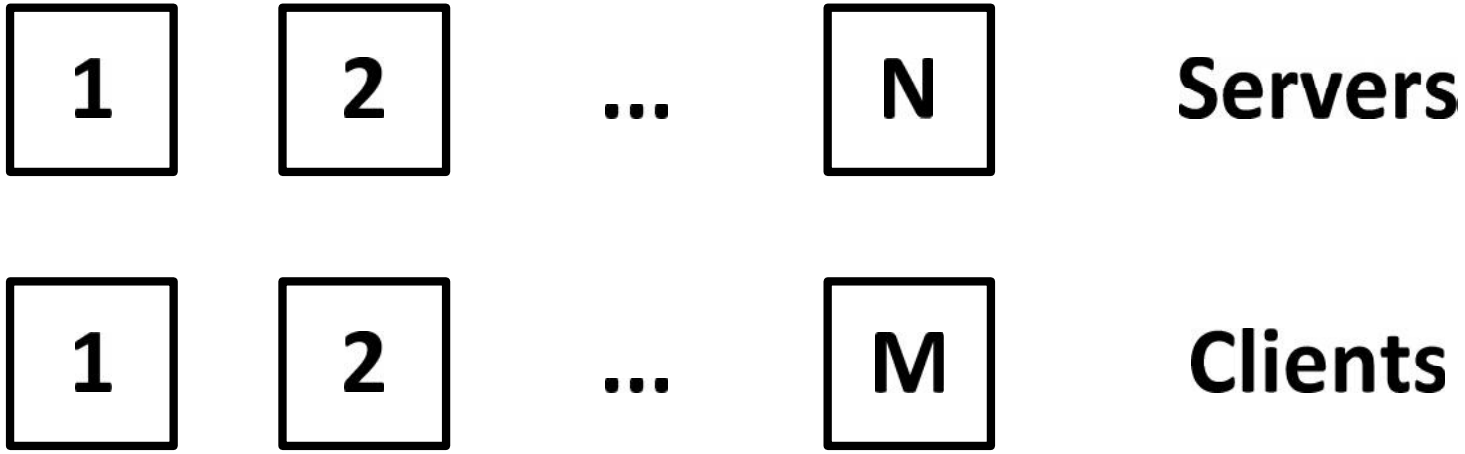
Redondance physique

- Réseau composé de M Clients et N Serveurs



- Haute disponibilité → Tolérance à la panne de N-1 Serveurs
- Load Balancing → Répartition de la charge sur les serveurs disponibles
- Fonctions essentielles sur internet → google, e-commerce, …
- Redondance logique ?

DNS round-robin (principe du tourniquet)

- Plusieurs adresses IP pour le même FQDN

```
nslookup
www.google.com
173.194.112.211
173.194.112.209
173.194.112.208
173.194.112.212
173.194.112.210
```

- Mécanisme simple et bon marché présentant quelques limites

DNS round-robin : fonctionnement

- La config. du serveur DNS ne tient pas compte de la disponibilité de chaque nœud IP

```
nslookup  lb.tdeig.ch
129.194.184.93 129.194.184.94 129.194.184.92
ping lb.tdeig.ch time out
```

- Le cache client peut diminuer l'efficacité du mécanisme de round-robin

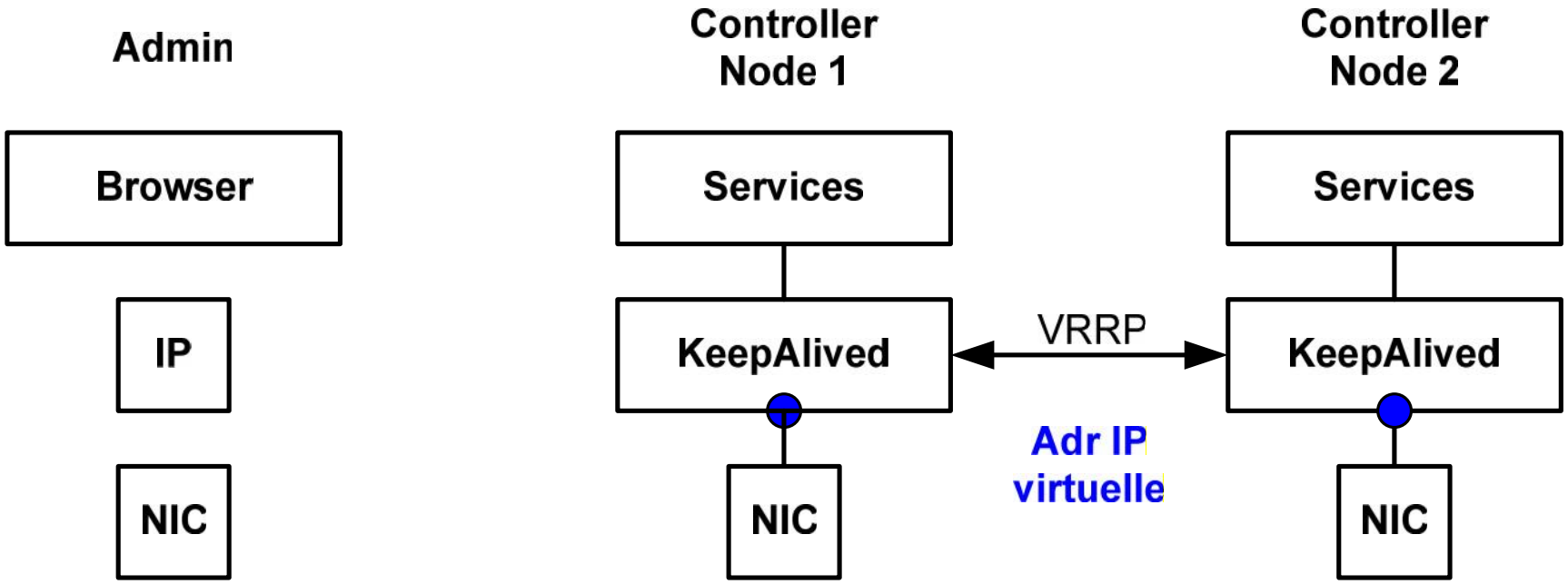
- Le basculement du client (navigateur, …, ping) en cas d'indisponibilité du serveur n'est pas garanti; il dépend de l'implémentation du client et demande souvent une action de l'utilisateur

GlusterFS (étudié dans le cours Virtualisation)

- Système de fichiers distribués
- Utilise un(des) FS natif(s) Linux = Ext4, …
- Volume est composé de briques (exemple = 1 brique/serveur)
- Elles sont montées (mount) du côté client
- Traducteur de distribution capable de répartir un fichiers sur N briques
- Traducteur de réplication capable de gérer la redondance
- Les performances sont proportionnelles au nombre de clients
- Astuce de GlusterFS : rendre le client intelligent
- Projet semestre → http://www.tdeig.ch/linux/Basbous_RPS.pdf

Disponibilité avec KeepAlived

- Objectif : garantir la disponibilité d'un nœud IP (aux administrateurs)

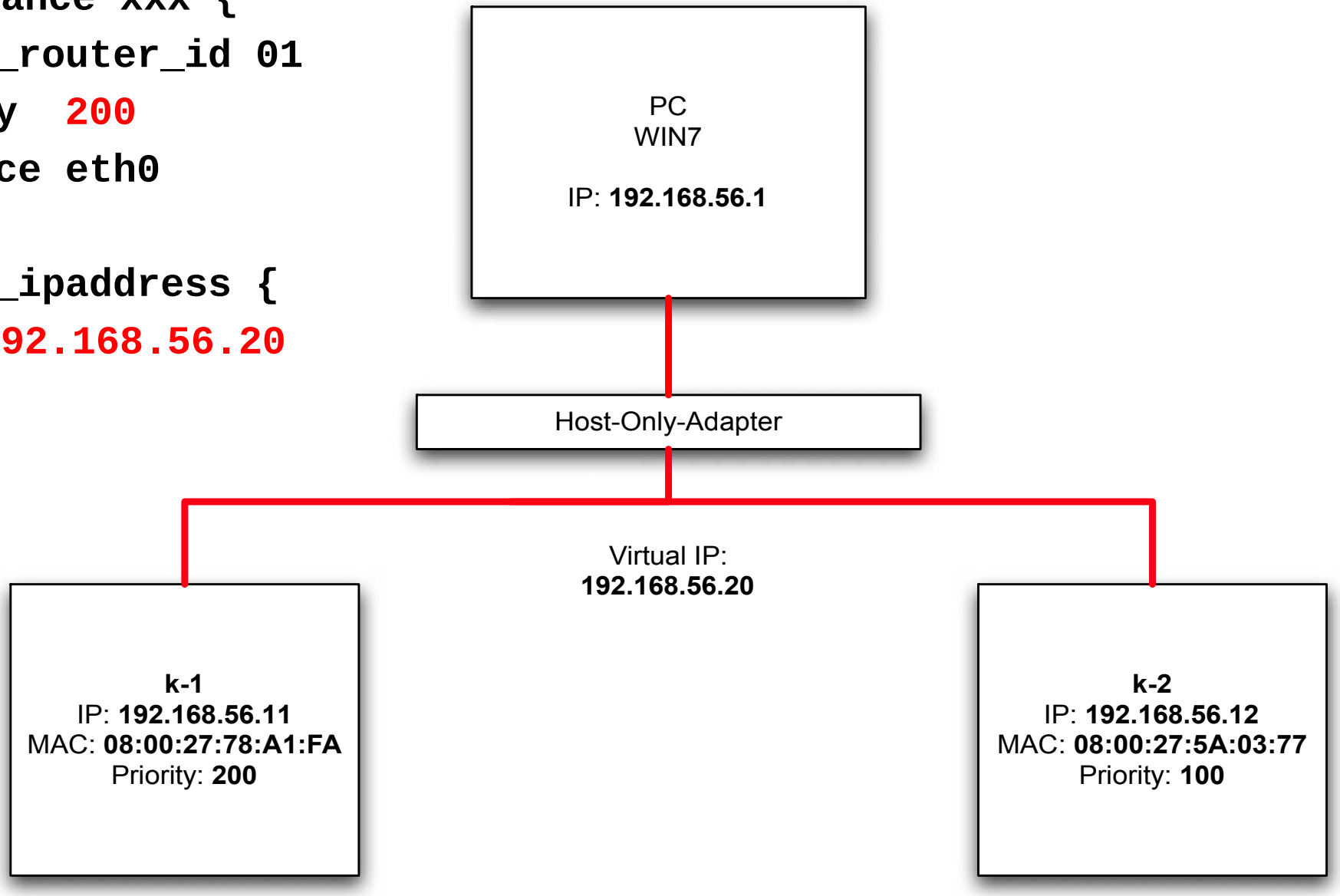


1. Le nœud avec la priorité supérieure est maître
2. Le maître envoie régulièrement un message multicast aux esclaves VRRP (rfc 5798) = Virtual Router Redundancy Protocol
3. L'esclave qui ne reçoit plus ces messages devient maître
4. Le client utilise l'adresse IP virtuelle

Labo §1 (30 min) : Disponibilité avec KeepAlived (1/2)

```
vrrp_instance xxx {
  virtual_router_id 01
  priority  200
  interface eth0

  virtual_ipaddress {
        192.168.56.20
  }
}
```



Labo §1 (30 min) : Disponibilité avec KeepAlived (2/2)

a) Préciser toutes les étapes dans un ordre logique

…

Démarrer le nœud le moins prioritaire

Analyser les logs

Utiliser ip addr show (à la place de ifconfig)

Démarrer l'autre nœud

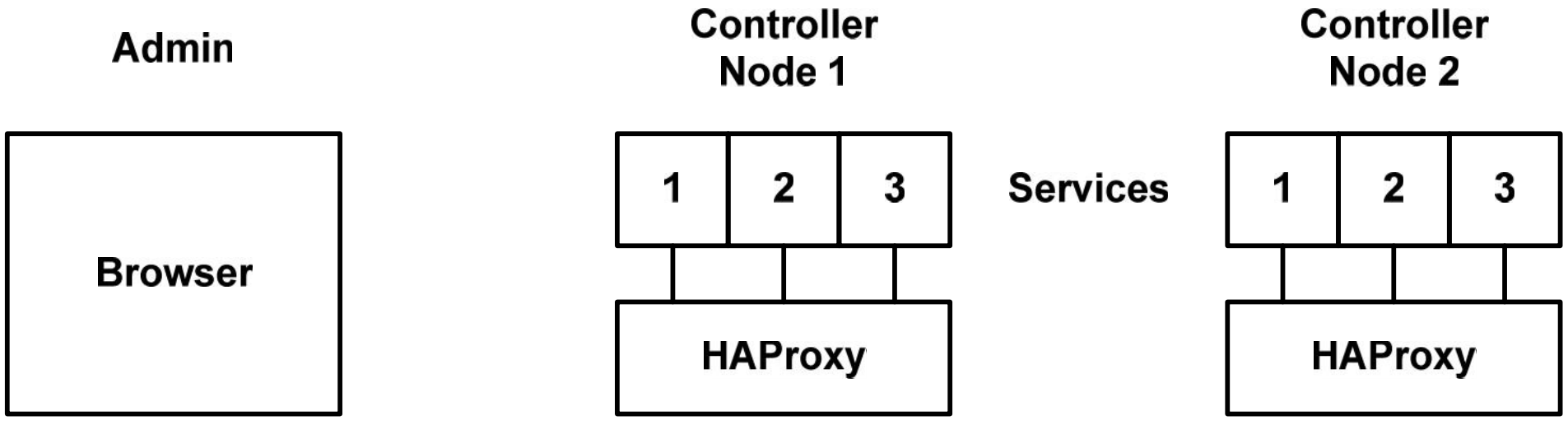
b) Etudier le basculement en déconnectant la VM

Des paquets sont-ils perdus ?

c) Etudier l'acquisition VRRP

Répartition de charge avec HAProxy

- Objectif : offrir la répartition de charge pour les services (d'administration)



1. Chaque service (http, …) est redondant
2. Le répartiteur de charge HAProxy utilise un mécanisme de rotation (round-robin)

- HAProxy → http://haproxy.1wt.eu/

HAProxy Configuration (haproxy.cfg)

```
listen name 192.168.56.10:80
```

Socket du Load Balancer côté Client

```
  balance  roundrobin
```

Algorithme de répartition

```
  mode http
```

Health check = http

```
  server  web-1 2.2.2.1:80  check
  server  web-2 2.2.2.2:80  check
```

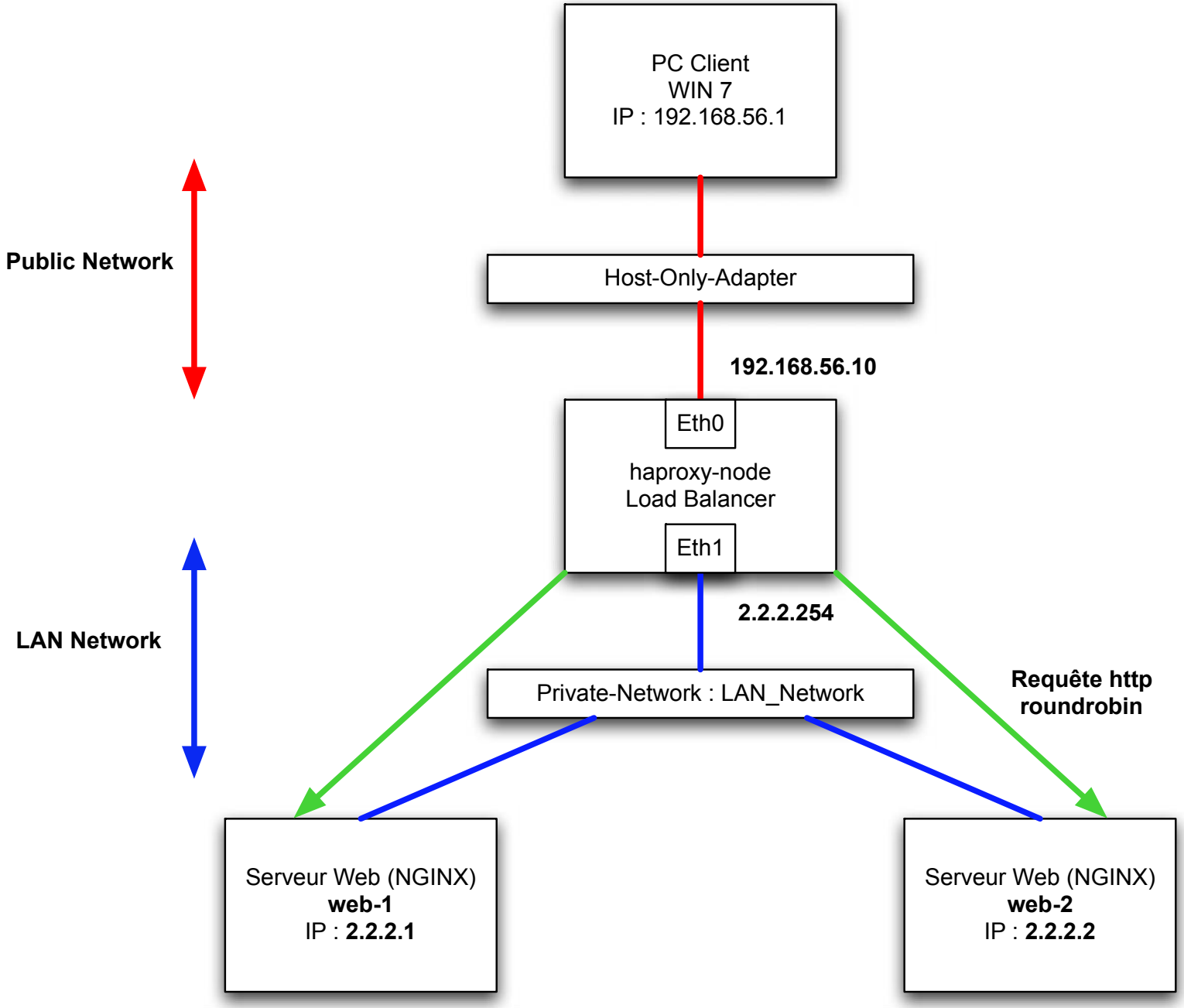
Serveurs web

```
listen  stats   192.168.56.10:8080
```

Socket pour lire les statistiques

http://www.haproxy.org/download/1.4/doc/configuration.txt

Labo §2 (40') : Répartition de charge avec HAProxy (1/2)



Labo §2 (40') : Répartition de charge avec HAProxy (2/2)

a) Etudier le fichier de config. /etc/haproxy/haproxy.cfg

b) Tester la répartition de charge
   Préciser toutes les étapes dans un ordre logique

c) Analyser les statistiques depuis http://192.168.56.10:8080/stats

d) Algorithmes supportés ?

e) Etudier 2 acquisitions Wireshark check_tcp et check_http

f) A quoi sert le paramètre forwardfor ?

Redondance et connexion persistante

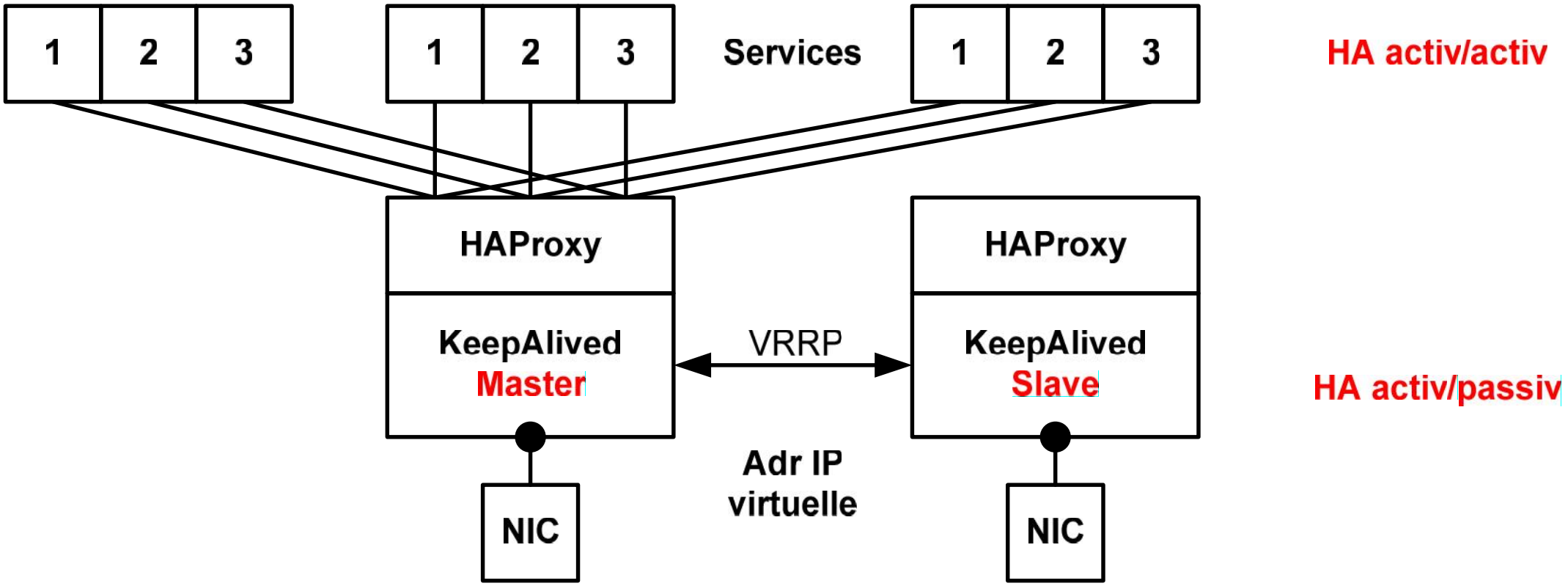
- L’architecture mise en œuvre convient pour un flux http sur des pages html statiques car le réponse sera la même quelque soit le serveur utilisé

- En cas de service orienté connexion (authentification de l’utilisateur, session SSL, …), toutes les requêtes doivent aboutir au même serveur

- On parle de connexion persistante réalisée par exemple en ajoutant un cookie

- Voir document Le Load Balancing pour les Nuls de Willy Tarreau

2 types de redondance mises en oeuvre



- HA activ/passiv (hot swapping)

  → un seul nœud actif + commutation en cas d'incident

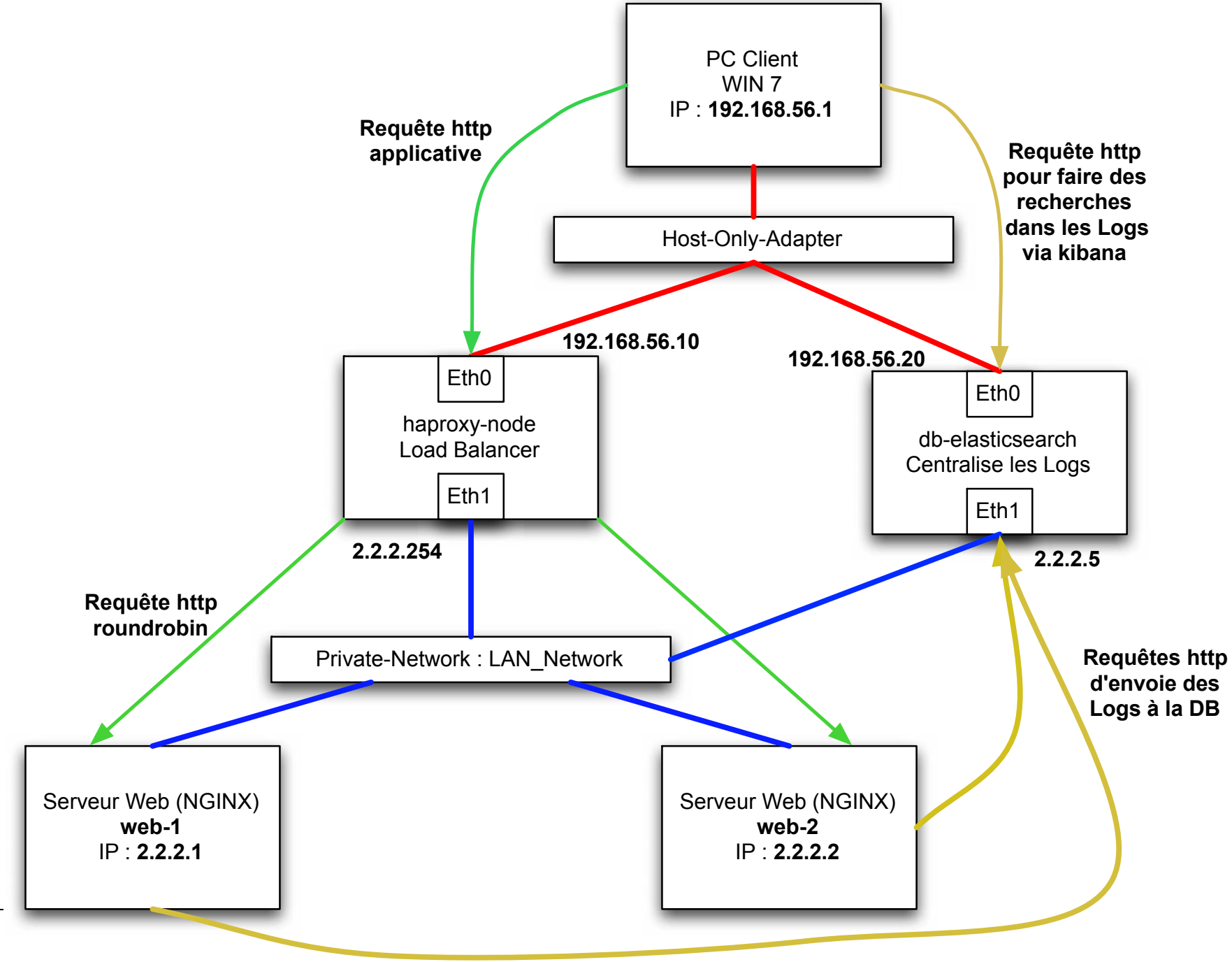
- HA activ/activ (load balancing)

  → répartition de charge entre les serveurs

Annexes du labo & Liens

- Pour_Les_Nuls par Willy Tarreau (concepteur HAProxy)
- Produit commercial Exceliance
- IBM Cloud
- TM Chalut 2014 → http://www.tdeig.ch/linux/Chalut_RTM.pdf
- Solution propriétaire : F5 BigIp avec Local Traffic Manager (90 kF)
  http://www.f5.com/pdf/products/big-ip-platforms-datasheet.pdf
  https://f5.com/products/modules/local-traffic-manager

Labo §3 (10 min) : Centralisation et analyse des logs (1/3)

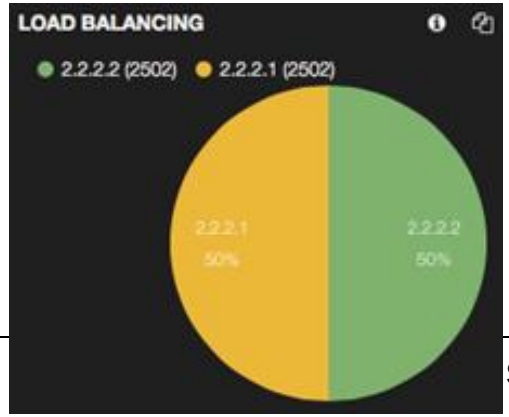


Labo §3 (10 min) : Centralisation et analyse des logs (2/3)

- 4 VMs à disposition dans le dossier Log_Part
- Log des serveurs web → **`/var/log/nginx/access.log`**

{"message":"192.168.56.1 2.2.2.1 29/Oct/2014:14:11:31 +0100 GET / HTTP/1.1 200 194 - Mozilla/5.0 (Windows N
","@version":"1","@timestamp":"2014-10-29T13:11:31.825Z","type":"web_nginx_access","host":"0.0.0.0","path...

- Service Logstash de conversion au format JSON (JAVA Script Object Notation)
- Centraliser les logs dans la base de données Elasticsearch
- Affichage web Kibana



Logs : Format JSON

```
"message"  =>  "Dec     3  00:24:38  controller  haproxy[1474]:    10.2.4.107:39510
[03/Dec/2013:00:24:38.457] nova-compute-api nova-compute-api/controller-2 0/0/4 280
-- 0/0/0/0/0 0/0",

              "@timestamp" => "2013-12-03T00:24:38",
                "@version" => "1",
                    "type" => "syslog_haproxy",
                    "path" => "/var/log/controller1/haproxy_access.log",
                 "program" => "haproxy",
                     "pid" => "1474",
             "received_at" => "Dec  3 00:24:38",
           "received_from" => "controller",
            "@source_host" => "controller",
                "@message" => "  10.2.4.107:39510  [03/Dec/2013:00:24:38.457]  nova-
compute-api nova-compute-api/controller-2 0/0/4 280 -- 0/0/0/0/0 0/0"
```

- Démo → http://www.cerny-online.com/cerny.js/demos/json-pretty-printing
- Compatible rfc3164 (timestamp, …)
- http://logstash.net/docs/1.3.2/tutorials/getting-started-simple

Théorème du CAP (1/2)

- Illustration à partir d'un service de bases de données (SQL), distribuées sur plusieurs nœuds (cluster)

- Un système distribué est régi par trois contraintes :

  http://citeseerx.ist.psu.edu

- **Consistency** (Consistence) : Avoir la même donnée sur tous les nœuds à l'instant t

- **Availability** (Disponibilité) : Les clients peuvent en permanence obtenir les données

- **Partitionnement Tolerance** (Résistance au morcèlement) : Le système fonctionne malgré des problèmes réseaux

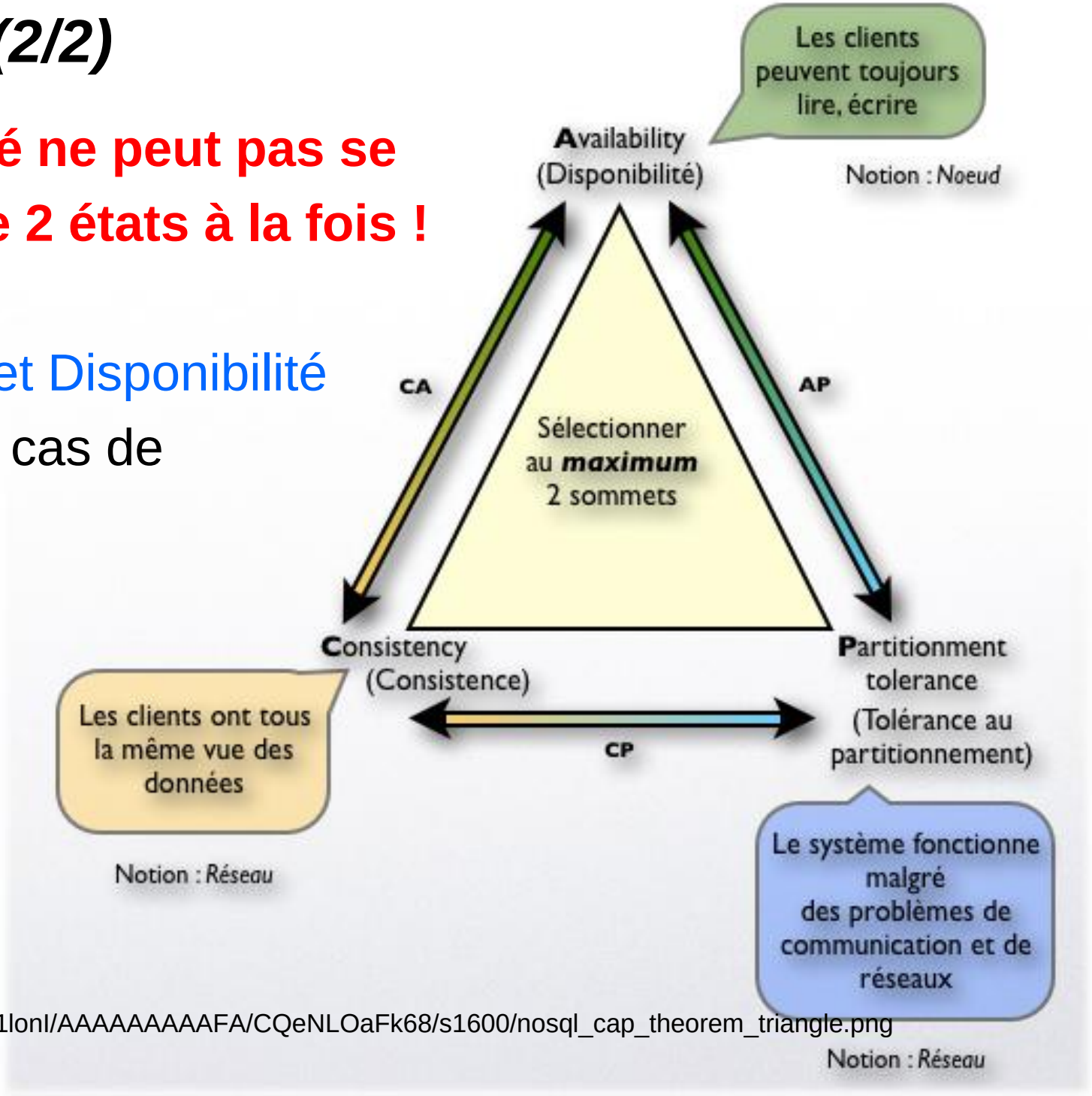
Théorème du CAP (2/2)

- **Un système distribué ne peut pas se trouver dans plus de 2 états à la fois !**

- Choix = Consistence et Disponibilité
  → devoir patienter en cas de problèmes réseau

- Si AP → Google

- Si CP → …



http://3.bp.blogspot.com/XfutdbkRTT0/Uek3LA1lonI/AAAAAAAAAFA/CQeNLOaFk68/s1600/nosql_cap_theorem_triangle.png

Supervision de l'infrastructure avec Shinken

- Illustration à partir du travail de diplôme de Huseyin Bilgin
  http://www.tdeig.ch/shinken/Bilgin_DEF.pdf

- Architecture & fonctionnement de Shinken (slides 14-20, 23,25)
- Méthodologie (slides 3-11)
- Scénario (slide 21)
- Résultats, affichages (slides 1,26-32)

- Mémoire → http://www.tdeig.ch/shinken/Bilgin_RTB.pdf