whoami

Alexander Penev
Email: alexander.penev@bytesource.net
Twitter: @apenev
@ByteSourceNet

JEE, Databases, Linux, TCP/IP

Fan of (automatic) testing, TDD, ADD, BDD…..

Like to design high available and scalable systems :-)

Zero Downtime Architectures

- Base on a customer project with the classic JEE Application Stack
    - Classic web applications with server side code
    - HTTP based APIs
- Goals, Concepts and Implementation Techniques
    - Constraints and limitations
- Developement guidelines
- How these concepts can be applied to the new cuttung edge technolgies
    - Single page Java Script based Apps
    - Mobile clients
    - Rest APIs
    - Node.js
    - NoSQL stores

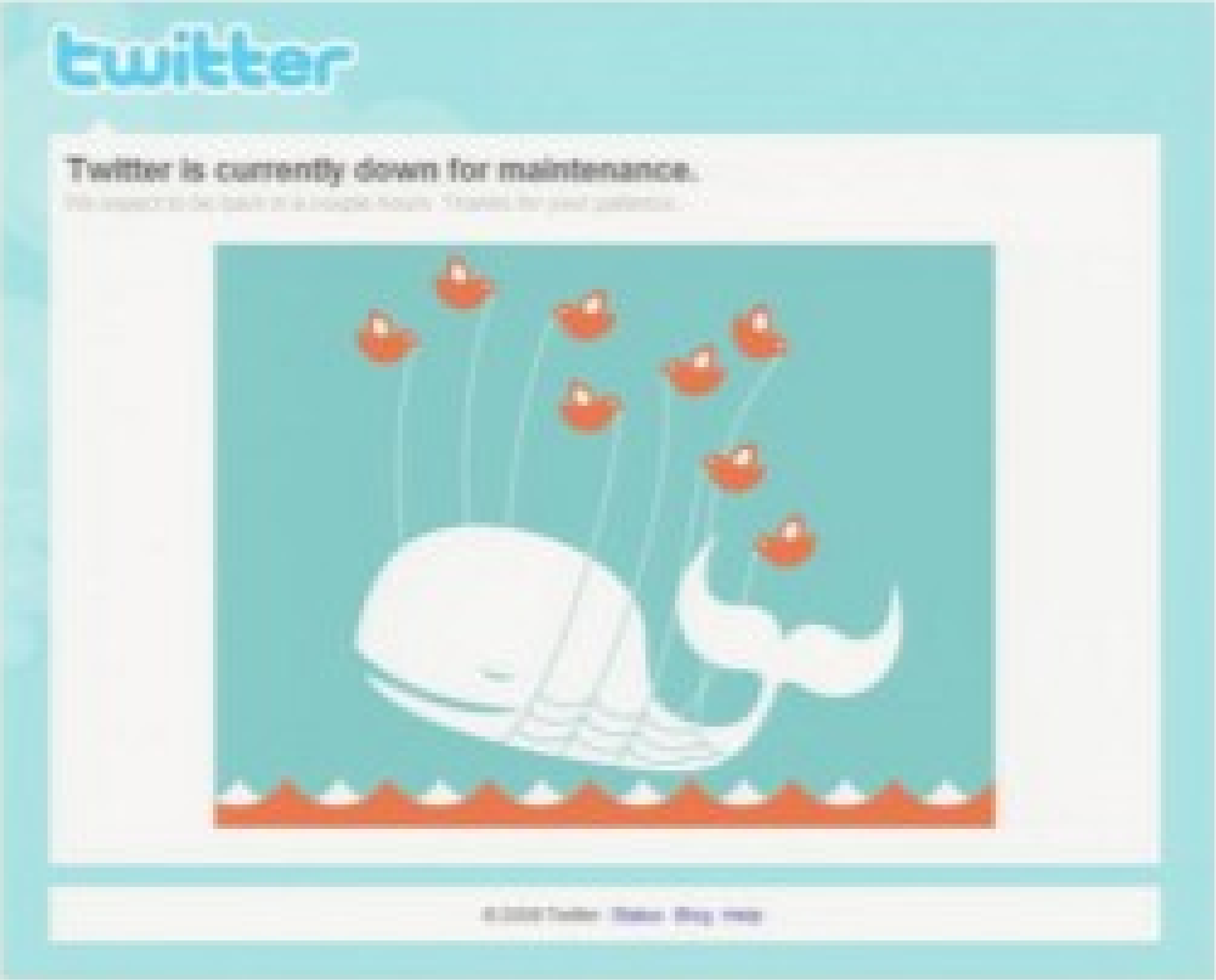
Zero Downtime Architecture?

- My database server has 99.999% uptime
- We have Tomcat cluster
- Redundant power supply
- Second Datacenter
- Load Balancer
- Distribute routes over OSPF
- Deploy my application online
- Second ISP
- Session Replication
- Monitoring
- Data Replication
- Auto restarts

bytesource

Zero Downtime architecture: our definition

The services from the end user point of view could be always available



twitter

Twitter is currently down for maintenance.

We expect to be back in a couple hours. Thanks for your patience.

© 2008 Twitter · Status · Blog · Help

bytesource

Our Vision

Identify all sources of downtime and remove all them

http://www.metalsucks.com/wp-content/uploads/2011/09/p360.jpg

When could we have a downtime (unplanned)?

- Human errors
- Server node has crashed
  - Power supply is broken, RAM Chip bumed out, OS just crashed
- Server Software just crashed
  - IO errors, software bug, tablespace full
- Network is unavailable
  - Router crashed, Uplink down
- Datacenter is down
  - Uplinks down (notorious bagger :-))
  - Flood/Fire
  - Aircondition broken
  - Hit by a nuke (not so often :-))

bytesource

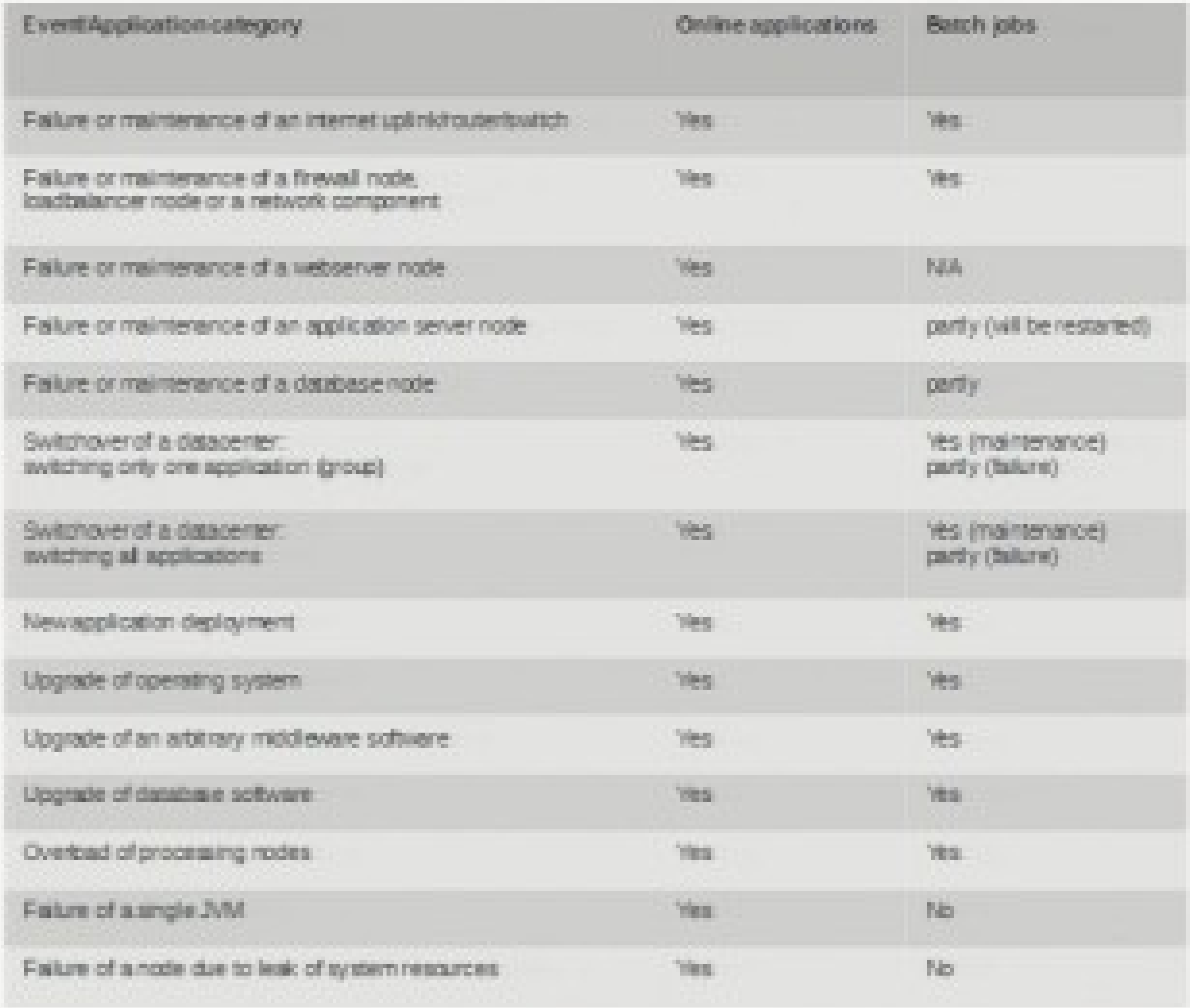
When could we need a downtime (planned)?

- Replace a hardware part
- Replace a router/switch
- Firmware upgrade
- Upgrade/exchange the storage
- Configuration of the connection pool
- Configuration of the cluster
- Upgrade the cluster software
- Recover from a logical data error
- **Upgrade the database software**
- **Deploy a new version of our software**
- **Move the application to another data center**

How can we avoid downtime

- Redunancy
  - Hardware, network
  - Uplinks
  - Datacenters
  - Software
- Monitoring
  - Detect exhausted resources before the application notices it
  - Detect a failed node and replace it
- Software design
  - Idempotent service calls
  - Backwards compatibility
  - Live releases
- Scalability
  - Scale on more load
  - Protect from attacks (e.g. DDoS)

Requirements for a Zero Downtime Architecture: handling of events of failure or maintenance



Event/Application category	Online applications	Batch jobs
Failure or maintenance of an internet uplink/router/switch	Yes	Yes
Failure or maintenance of a firewall node, loadbalancer node or a network component	Yes	Yes
Failure or maintenance of a webserver node	Yes	N/A
Failure or maintenance of an application server node	Yes	partly (will be restarted)
Failure or maintenance of a database node	Yes	partly
Switchover of a datacenter: switching only one application (group)	Yes	Yes (maintenance) partly (failure)
Switchover of a datacenter: switching all applications	Yes	Yes (maintenance) partly (failure)
New application deployment	Yes	Yes
Upgrade of operating system	Yes	Yes
Upgrade of an arbitrary middleware software	Yes	Yes
Upgrade of database software	Yes	Yes
Overload of processing nodes	Yes	Yes
Failure of a single JVM	Yes	No
Failure of a node due to leak of system resources	Yes	No

Our goals and constraints

- Reduce downtime to 0
- Keep the costs low
  - No expensive propriatery hardware
  - Minimize the potential application changes/rewrites

http://www.signwarehouse.com/blog/how-to-keep-fixed-costs-low/

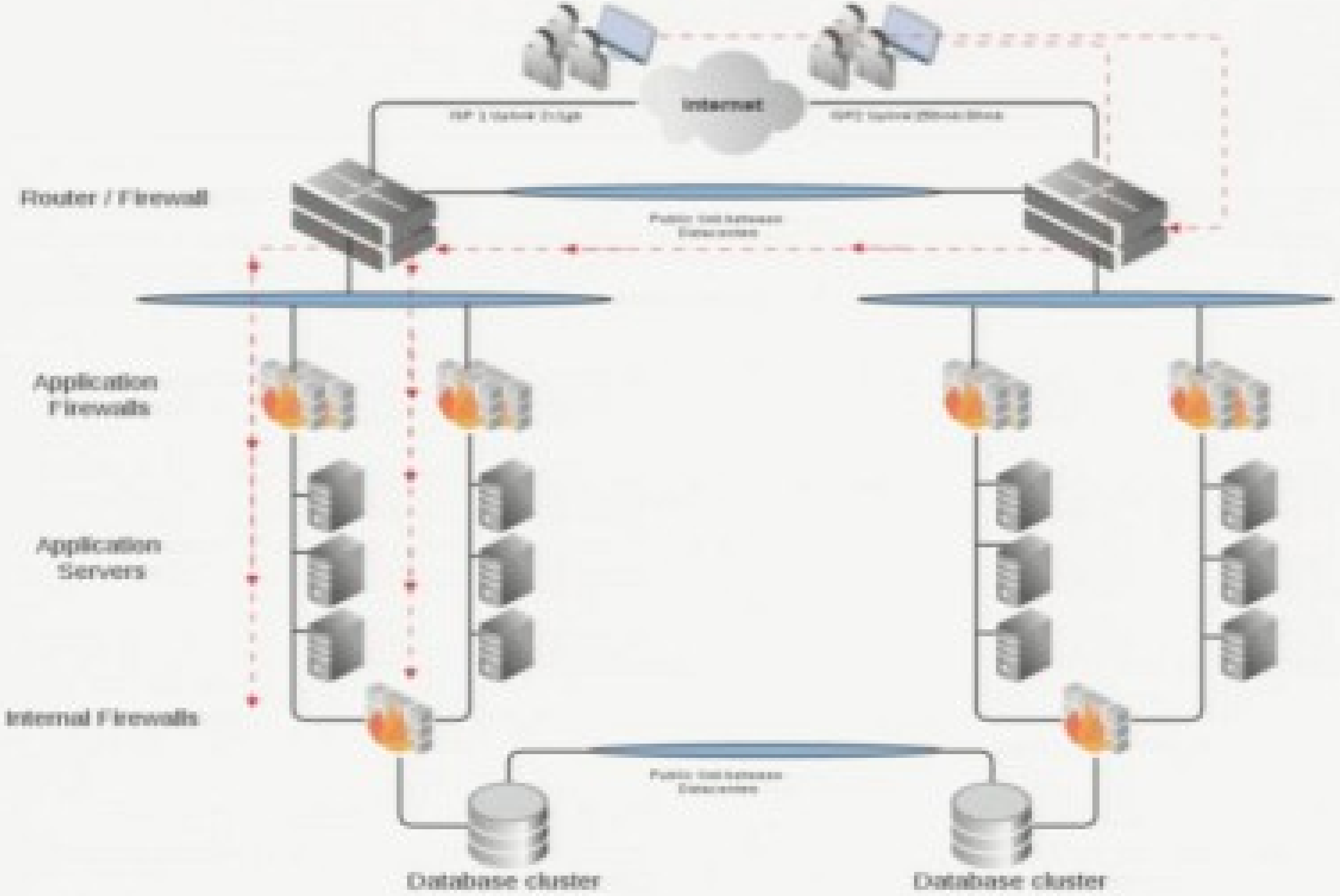
Our Concepts 1/4

- Independent Applications or Application Groups
  - One Application (Group) = IP Address
  - Communication between Application exclusively over this IP Address!

[illegible]

Our Concepts 2/4

Treat the internet and internal traffic independently



bytesource

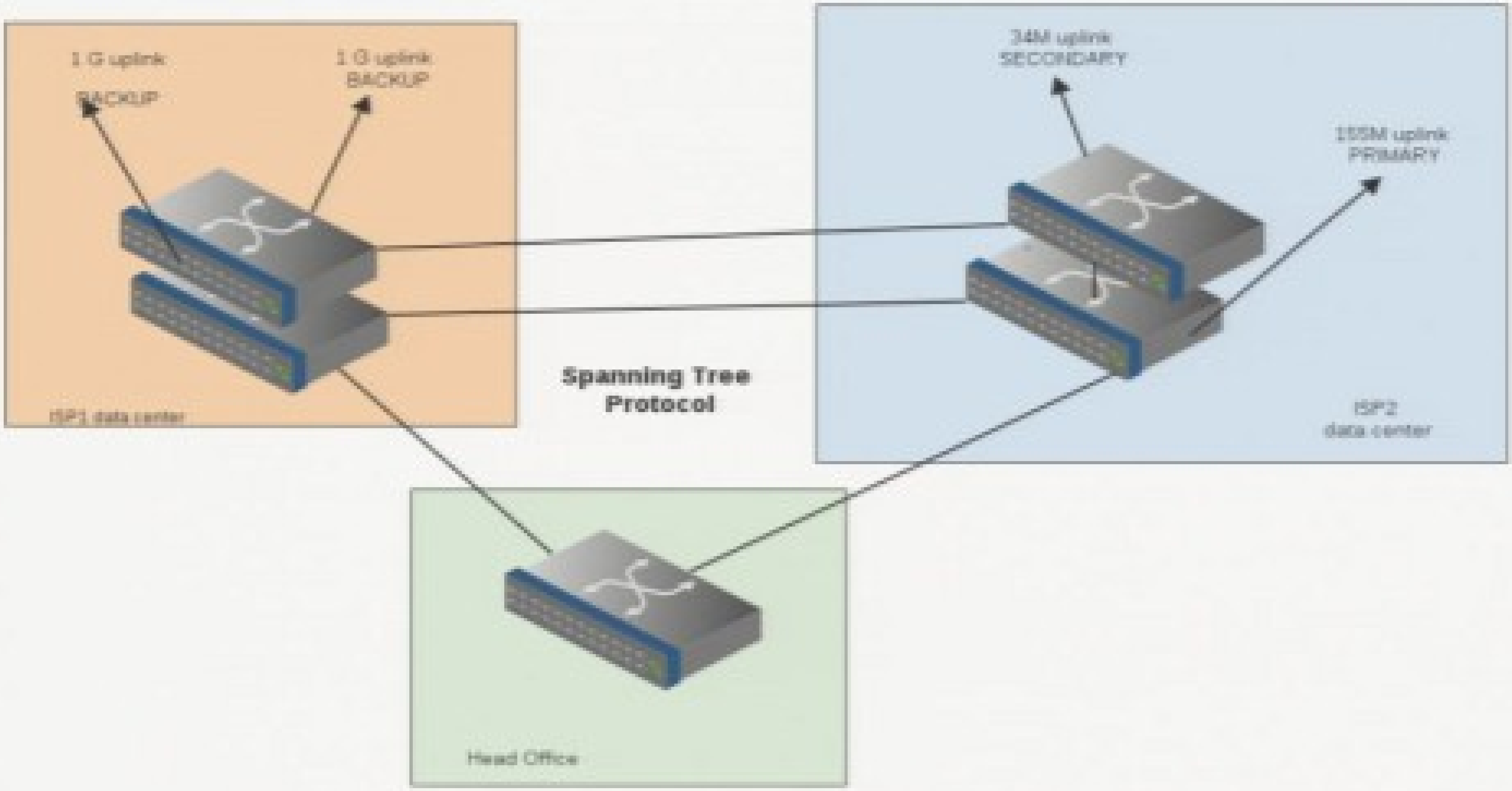
Our Concepts 3/4

- Reduce the downtime within a datacenter to 0
  - High available network
  - Redundant firewalls and load balancers
  - Web server farms
  - Application server clusters with sesion replication
  - Oracle RAC Cluster
  - **Downtime free application deployments**

Our Concepts 4/4

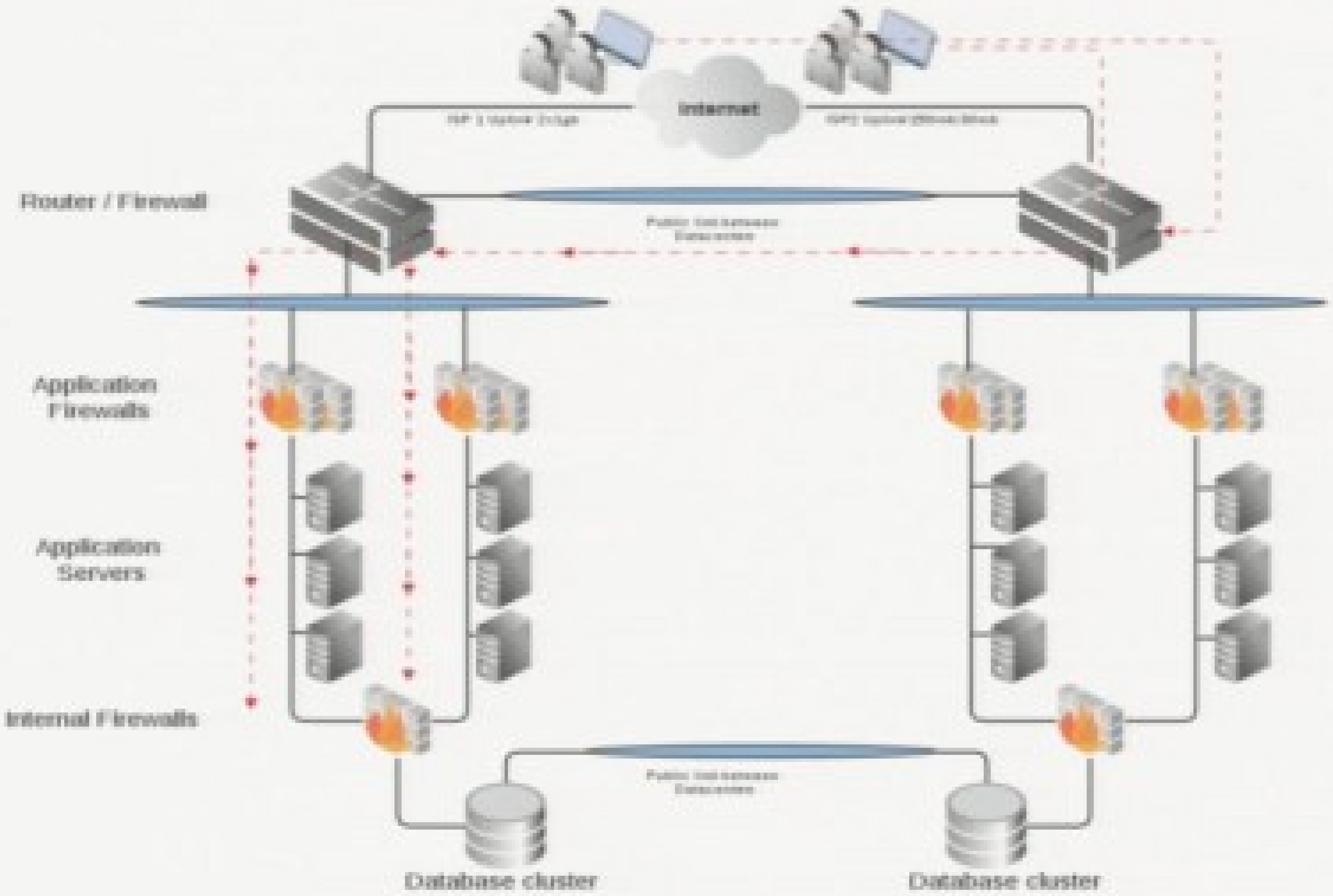
- Replicate the data on both datacenters
- and make the applications switchable

Implementation: Network (Layer 2)



Our Concepts 2/4

Treat the internet and internal traffic independently

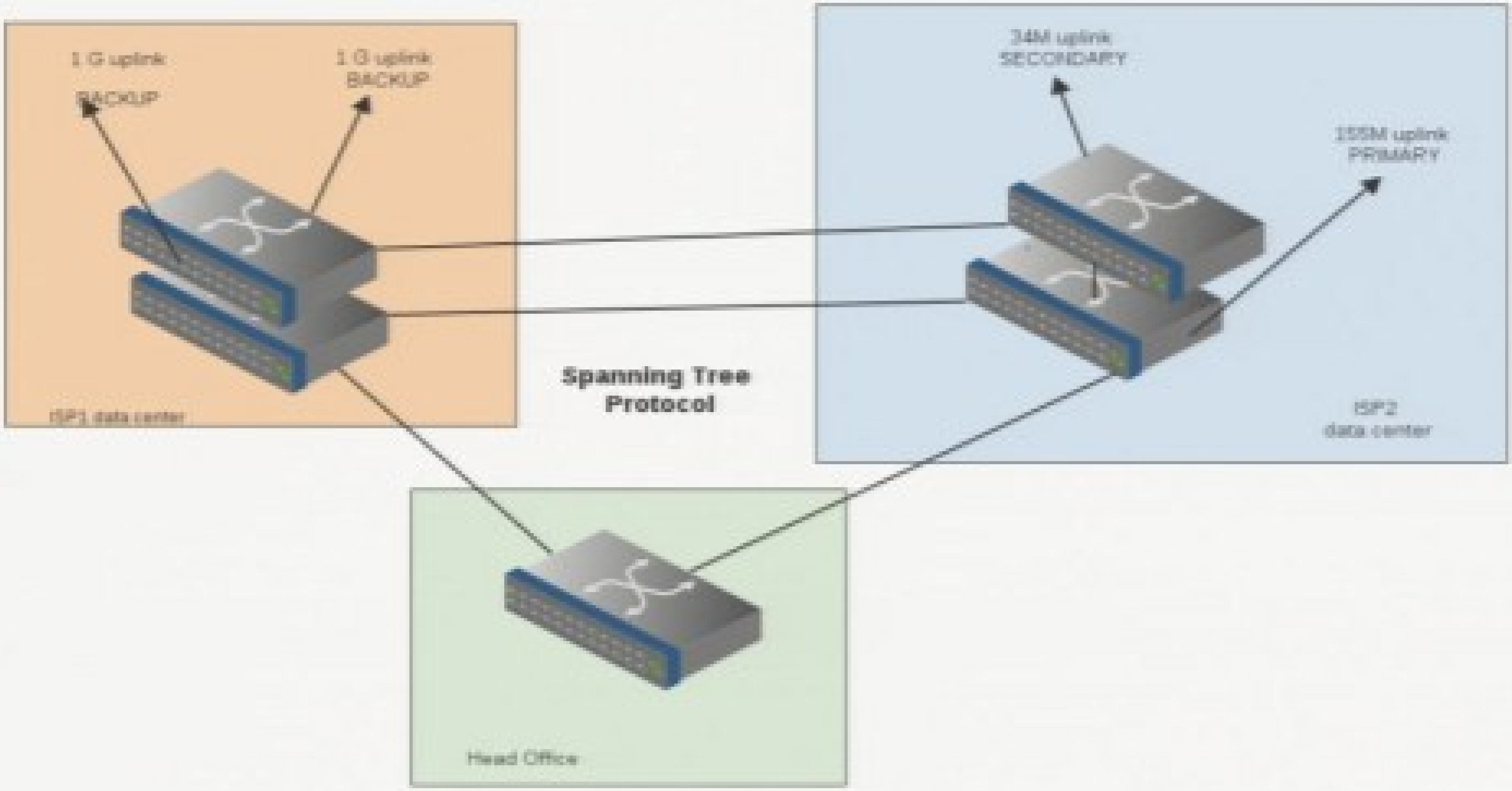


bytesource

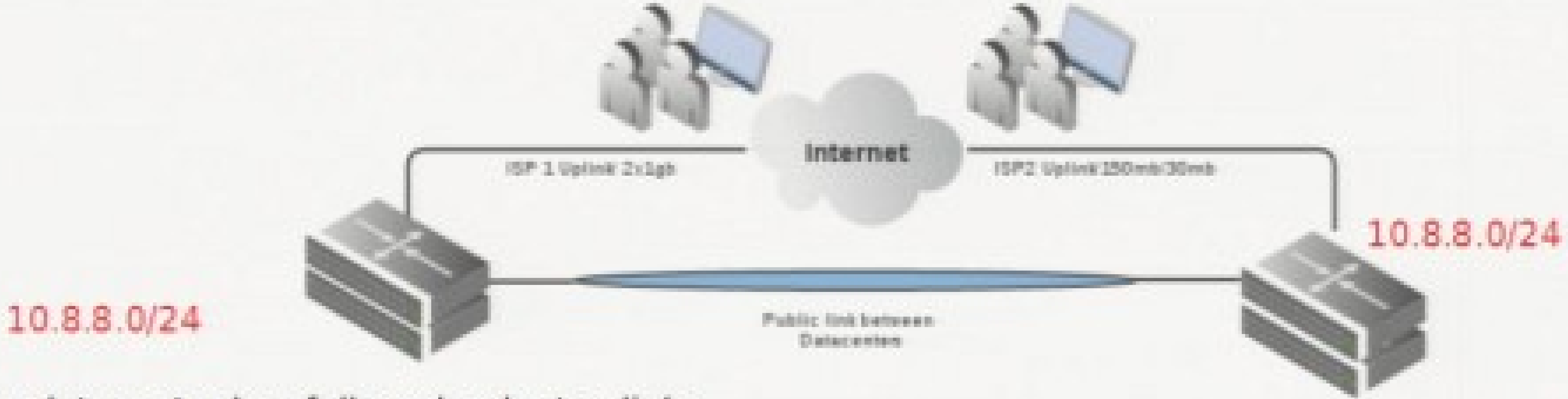
Our Concepts 3/4

- Reduce the downtime within a datacenter to 0
  - High available network
  - Redundant firewalls and load balancers
  - Web server farms
  - Application server clusters with sesion replication
  - Oracle RAC Cluster
  - **Downtime free application deployments**

Implementation: Network (Layer 2)

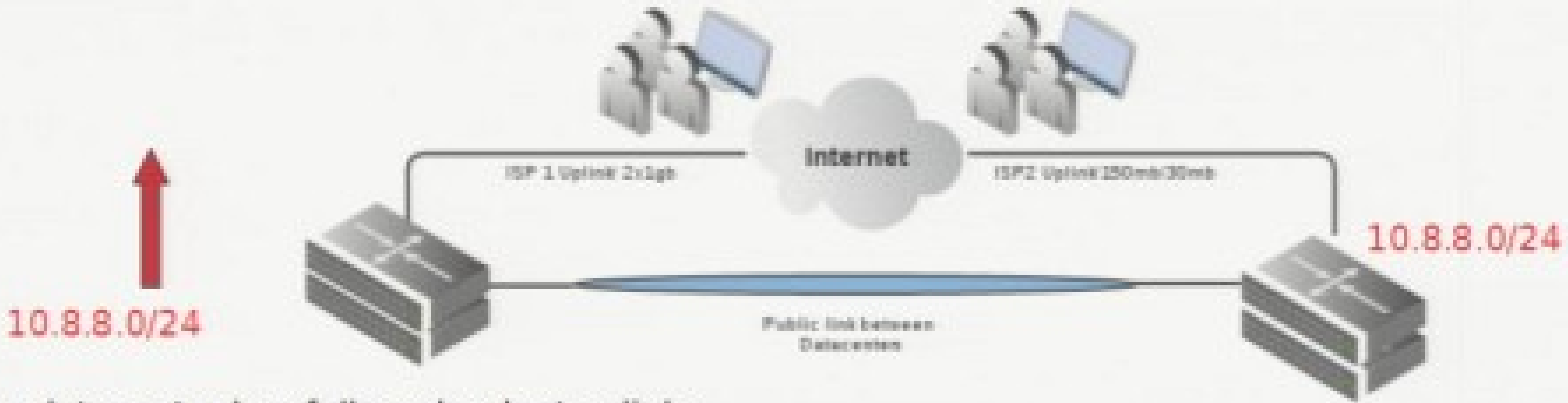


Concepts: Internet traffic, BGP(Border Gateway Protocol) 1/2



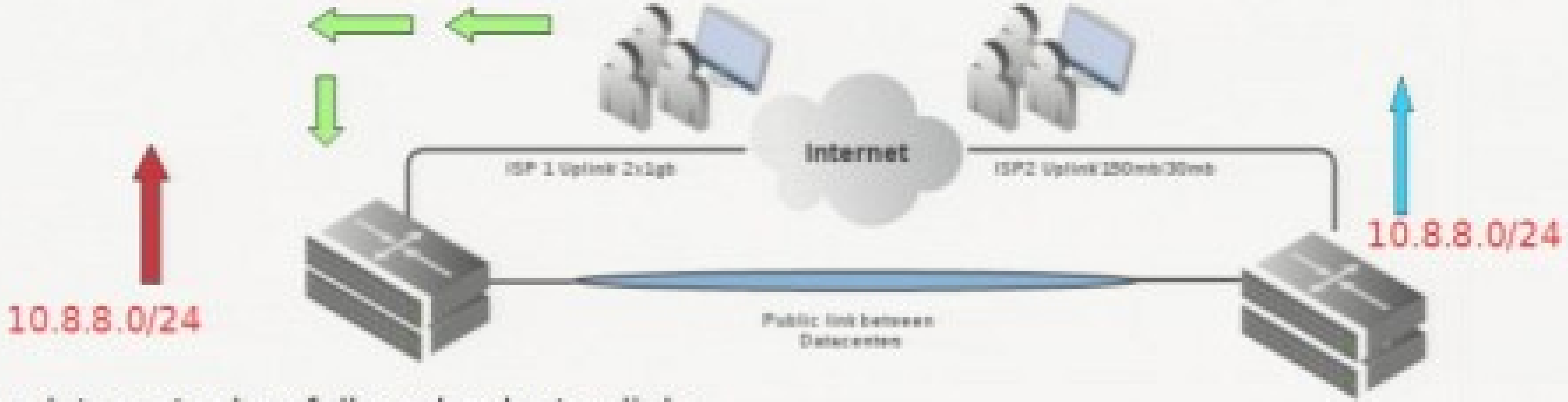
- Every datacenter has fully redundant uplinks
- **Own provider independent IP address range** (assigned by RIPE)
  - Hard to get in the moment (but not impossible)
- Propagate these addresses to the rest of the internet through both ISPs using BGP
  - Both DCs our addresses
  - The network path of one announcement could be preferred (for costs reasons)
- Switch of internet traffic
  - Gracefully by changing the preferences of the announcements
    - No single TCP session lost
  - In case of disaster the backup route is propagated automatically within seconds to minutes (depending on the internet distance)
- Protect us from connectivity problems between our ISPs and our customer ISPs

Concepts: Internet traffic, BGP(Border Gateway Protocol) 1/2



- Every datacenter has fully redundant uplinks
- **Own provider independent IP address range** (assigned by RIPE)
  - Hard to get in the moment (but not impossible)
- Propagate these addresses to the rest of the internet through both ISPs using BGP
  - Both DCs our addresses
  - The network path of one announcement could be preferred (for costs reasons)
- Switch of internet traffic
  - Gracefully by changing the preferences of the announcements
    - No single TCP session lost
  - In case of disaster the backup route is propagated automatically within seconds to minutes (depending on the internet distance)
- Protect us from connectivity problems between our ISPs and our customer ISPs

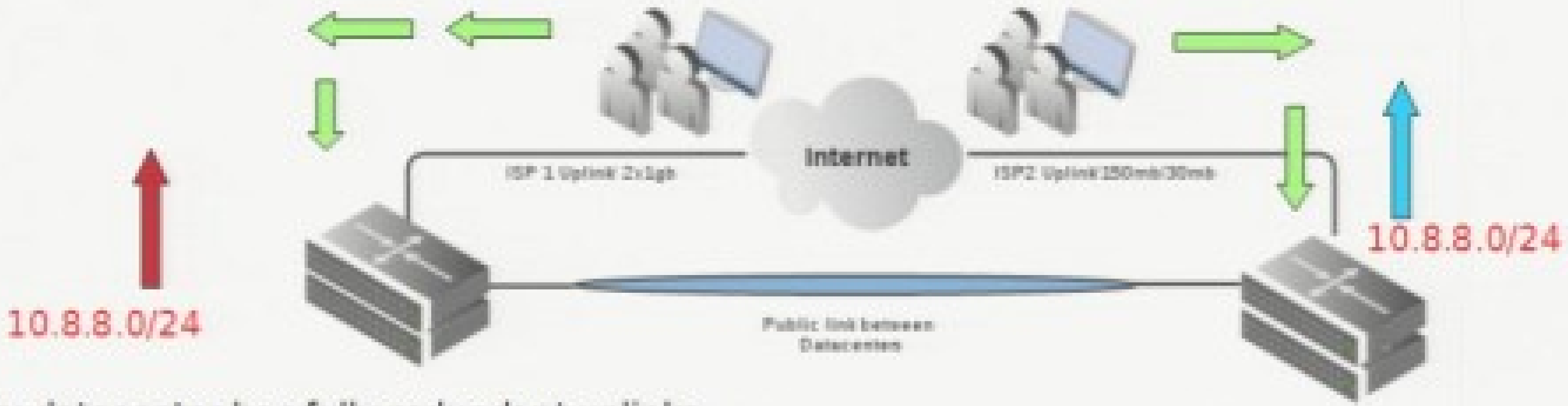
Concepts: Internet traffic, BGP(Border Gateway Protocol) 1/2



- Every datacenter has fully redundant uplinks
- **Own provider independent IP address range** (assigned by RIPE)
  - Hard to get in the moment (but not impossible)
- Propagate these addresses to the rest of the internet through both ISPs using BGP
  - Both DCs our addresses
  - The network path of one announcement could be preferred (for costs reasons)
- Switch of internet traffic
  - Gracefully by changing the preferences of the announcements
    - No single TCP session lost
  - In case of disaster the backup route is propagated automatically within seconds to minutes (depending on the internet distance)
- Protect us from connectivity problems between our ISPs and our customer ISPs

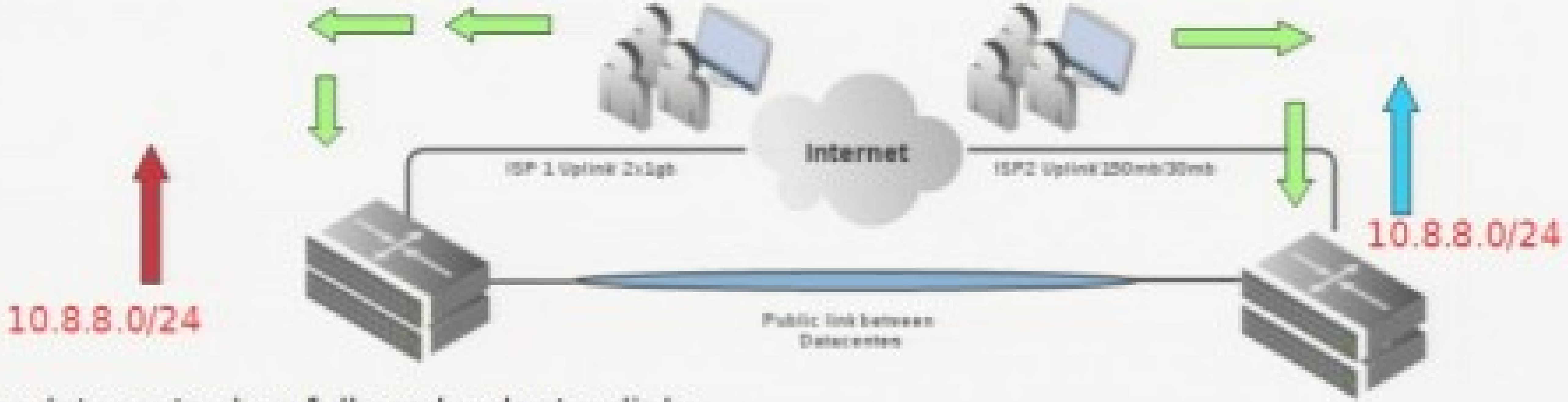
bytesource

Concepts: Internet traffic, BGP(Border Gateway Protocol) 1/2



- Every datacenter has fully redundant uplinks
- **Own provider independent IP address range** (assigned by RIPE)
  - Hard to get in the moment (but not impossible)
- Propagate these addresses to the rest of the internet through both ISPs using BGP
  - Both DCs our addresses
  - The network path of one announcement could be preferred (for costs reasons)
- Switch of internet traffic
  - Gracefully by changing the preferences of the announcements
    - No single TCP session lost
  - In case of disaster the backup route is propagated automatically within seconds to minutes (depending on the internet distance)
- Protect us from connectivity problems between our ISPs and our customer ISPs

Concepts: Internet traffic, BGP(Border Gateway Protocol) 1/2



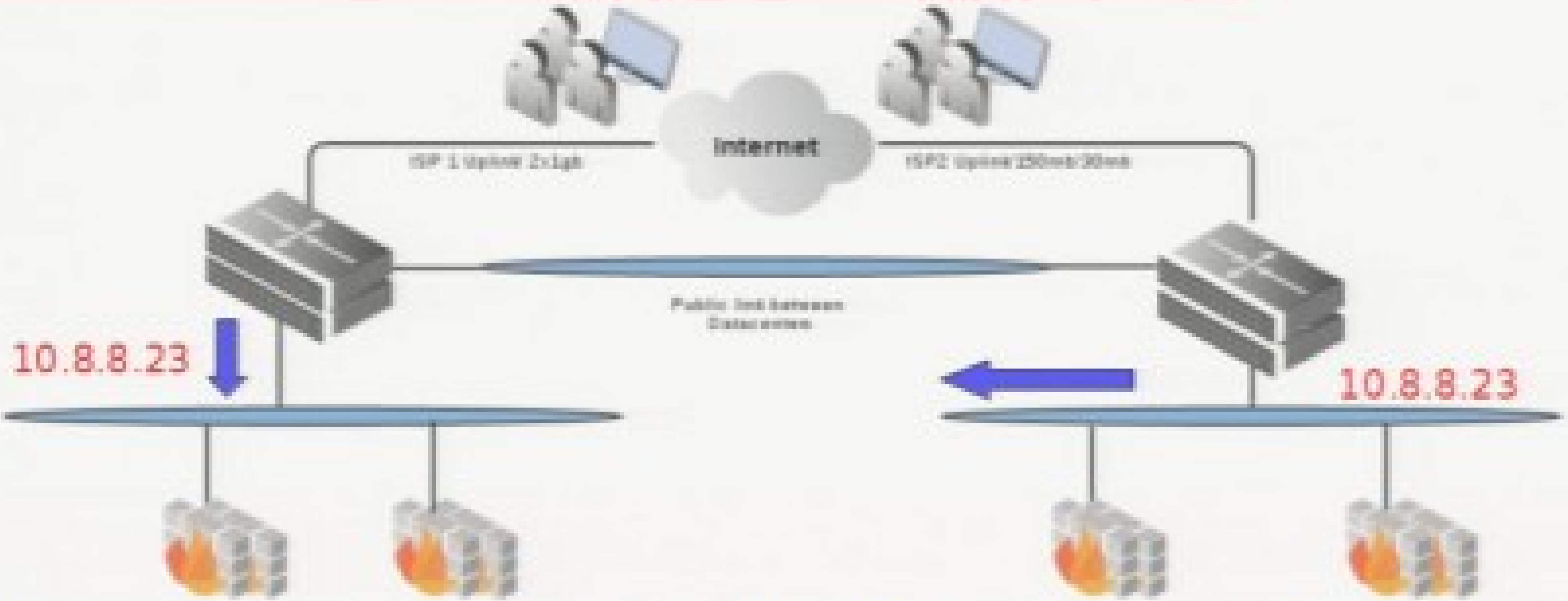
- Every datacenter has fully redundant uplinks
- **Own provider independent IP address range** (assigned by RIPE)
  - Hard to get in the moment (but not impossible)
- Propagate these addresses to the rest of the internet through both ISPs using BGP
  - Both DCs our addresses
  - The network path of one announcement could be preferred (for costs reasons)
- Switch of internet traffic
  - Gracefully by changing the preferences of the announcements
    - No single TCP session lost
  - In case of disaster the backup route is propagated automatically within seconds to minutes (depending on the internet distance)
- Protect us from connectivity problems between our ISPs and our customer ISPs

bytesource

Concepts: Internet traffic, use DNS ? 2/2

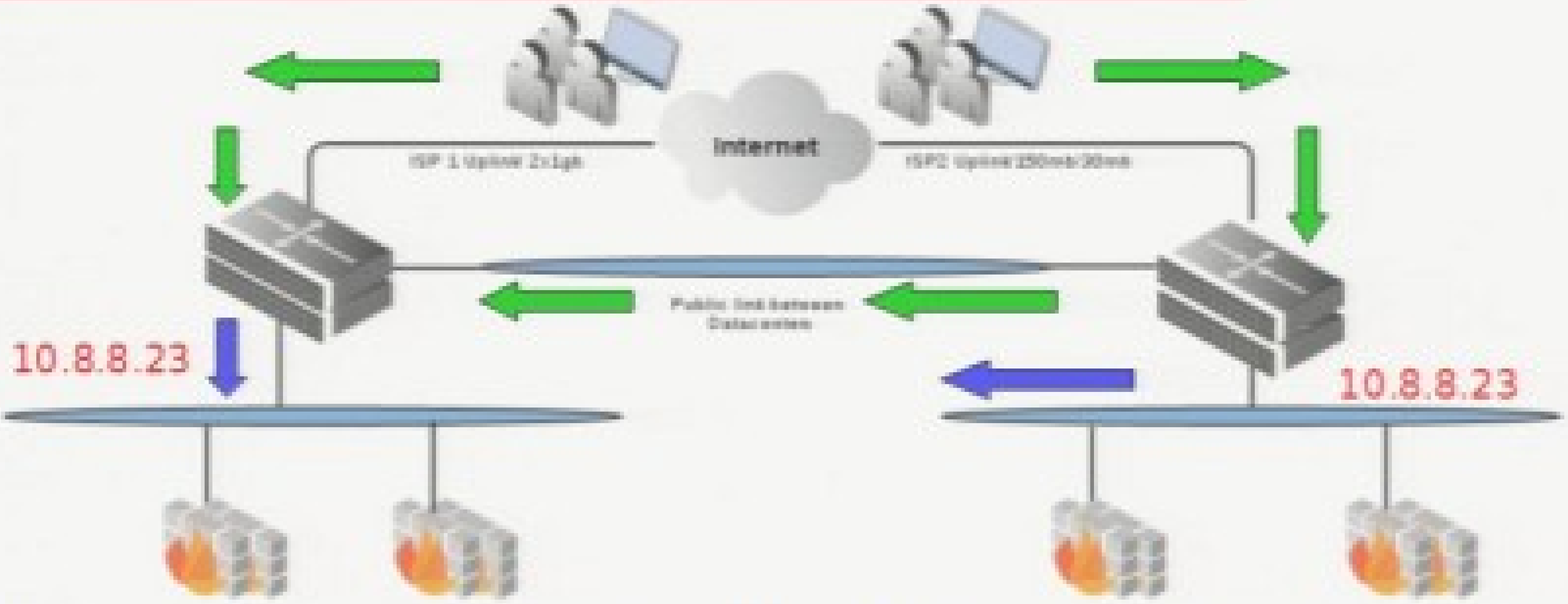
- **We don't use DNS for switching**
  - A datacenter switch based on DNS could take up to months to reach all customers and their software (e.g. JVMs caching DNS entries, default behaviour)
  - No need to restart browsers, applications and proxies on the customer site. The customer doesn't see any change at all (except that route to us has changed)
- **DNS is good for load balancing but not for High Availability!**

Concepts: Internal traffic



- OSPF (Open Shortest Path First) protocol for dynamic routing
  - Deals with redundant paths completely transparently
  - Can also do load balancing
- The second level firewalls (in front of the load balancers) announce the address to the rest of the routers
  - To switch the processing of a service, it's firewall just has to announce the route (could be also a /32) with a higher priority, after a second the traffic goes through the new route.
- Could be also used for a unattended switch of the whole datacenter
  - Just announce the same IPs from both sites with different priorities
  - If the one datacenter dies there are only announcements from the other one

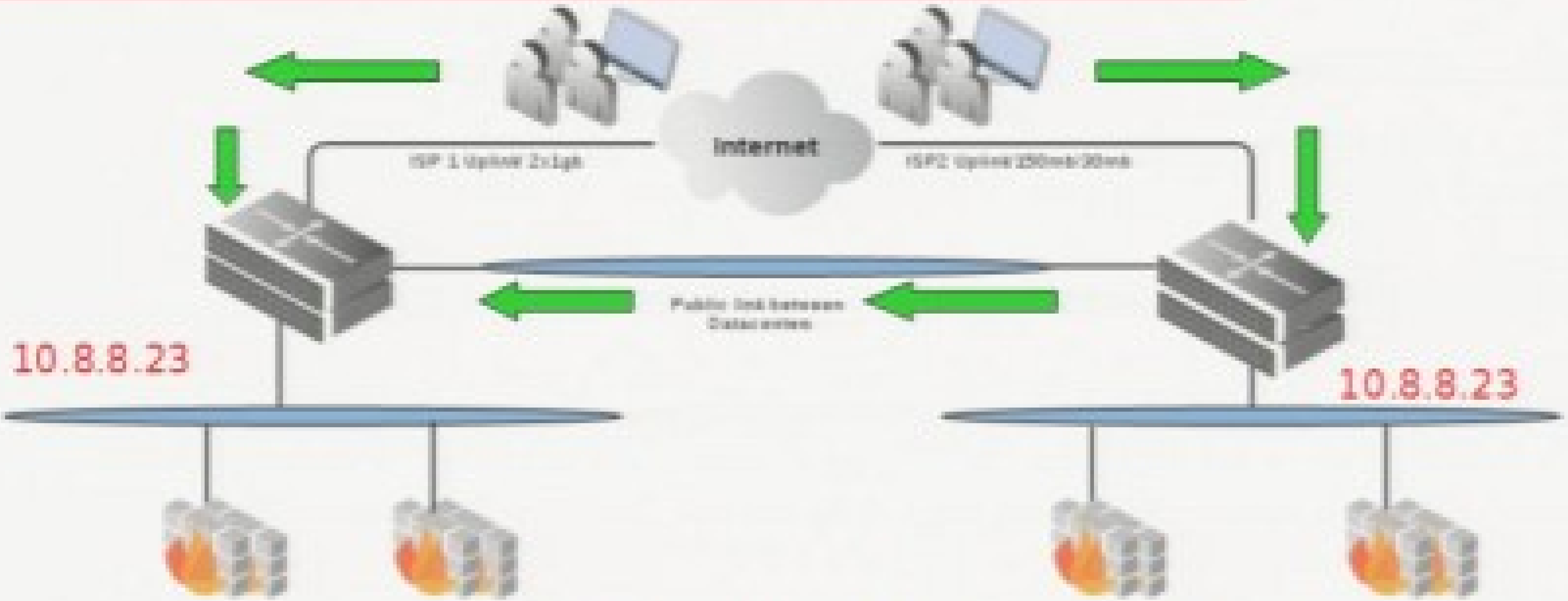
Concepts: Internal traffic



- OSPF (Open Shortest Path First) protocol for dynamic routing
  - Deals with redundant paths completely transparently
  - Can also do load balancing
- The second level firewalls (in front of the load balancers) announce the address to the rest of the routers
  - To switch the processing of a service, it's firewall just has to announce the route (could be also a /32) with a higher priority, after a second the traffic goes through the new route.
- Could be also used for a unattended switch of the whole datacenter
  - Just announce the same IPs from both sites with different priorities
  - If the one datacenter dies there are only announcements from the other one

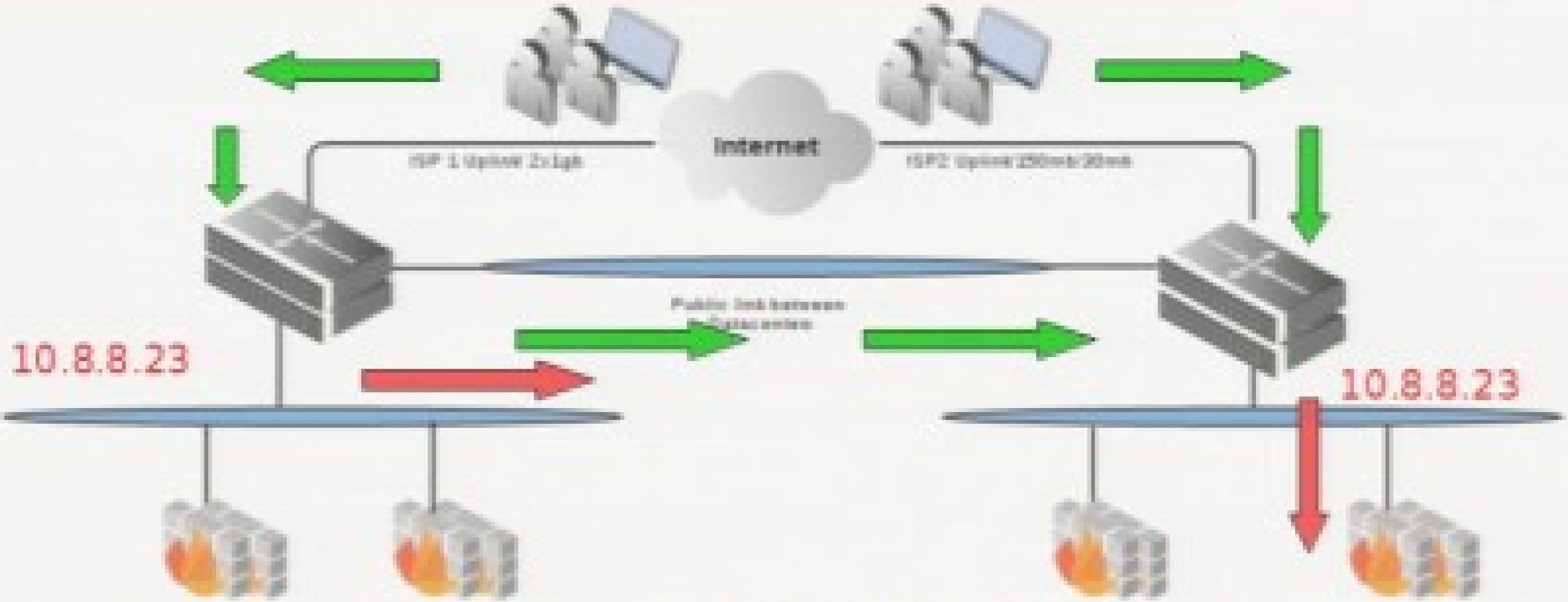
bytesource

Concepts: Internal traffic



- OSPF (Open Shortest Path First) protocol for dynamic routing
  - Deals with redundant paths completely transparently
  - Can also do load balancing
- The second level firewalls (in front of the load balancers) announce the address to the rest of the routers
  - To switch the processing of a service, it's firewall just has to announce the route (could be also a /32) with a higher priority, after a second the traffic goes through the new route.
- Could be also used for a unattended switch of the whole datacenter
  - Just announce the same IPs from both sites with different priorities
  - If the one datacenter dies there are only announcements from the other one

Concepts: Internal traffic



- OSPF (Open Shortest Path First) protocol for dynamic routing
  - Deals with redundant paths completely transparently
  - Can also do load balancing
- The second level firewalls (in front of the load balancers) announce the address to the rest of the routers
  - To switch the processing of a service, it's firewall just has to announce the route (could be also a /32) with a higher priority, after a second the traffic goes through the new route.
- Could be also used for a unattended switch of the whole datacenter
  - Just announce the same IPs from both sites with different priorities
  - If the one datacenter dies there are only announcements from the other one

Our Concepts

- ✓ Independent Applications or Application Groups
- ✓ Independent Internet and internal network trafic
- Reduce Downtime within a DC
- Replicate the data between the Dcs and make the application switchable

Zero Downtime within a datacenter

- High Available network
  - Redundant switches
    - Again using Spanning Tree Protocol
  - Redundant firewalls, routers, load balancers
    - Active/Passive Clusters
    - VRRP protocol implemeneted by keepalived
    - IP tables with contractd
- Web Server Apache farms
  - Managed by load balancer
- Application Server Cluster
  - Weblogic Cluster
  - With Session replication,
  - automcatic retries and restarts
- Oracle RAC database cluster
- Deployment without downtime

LVS load balancer

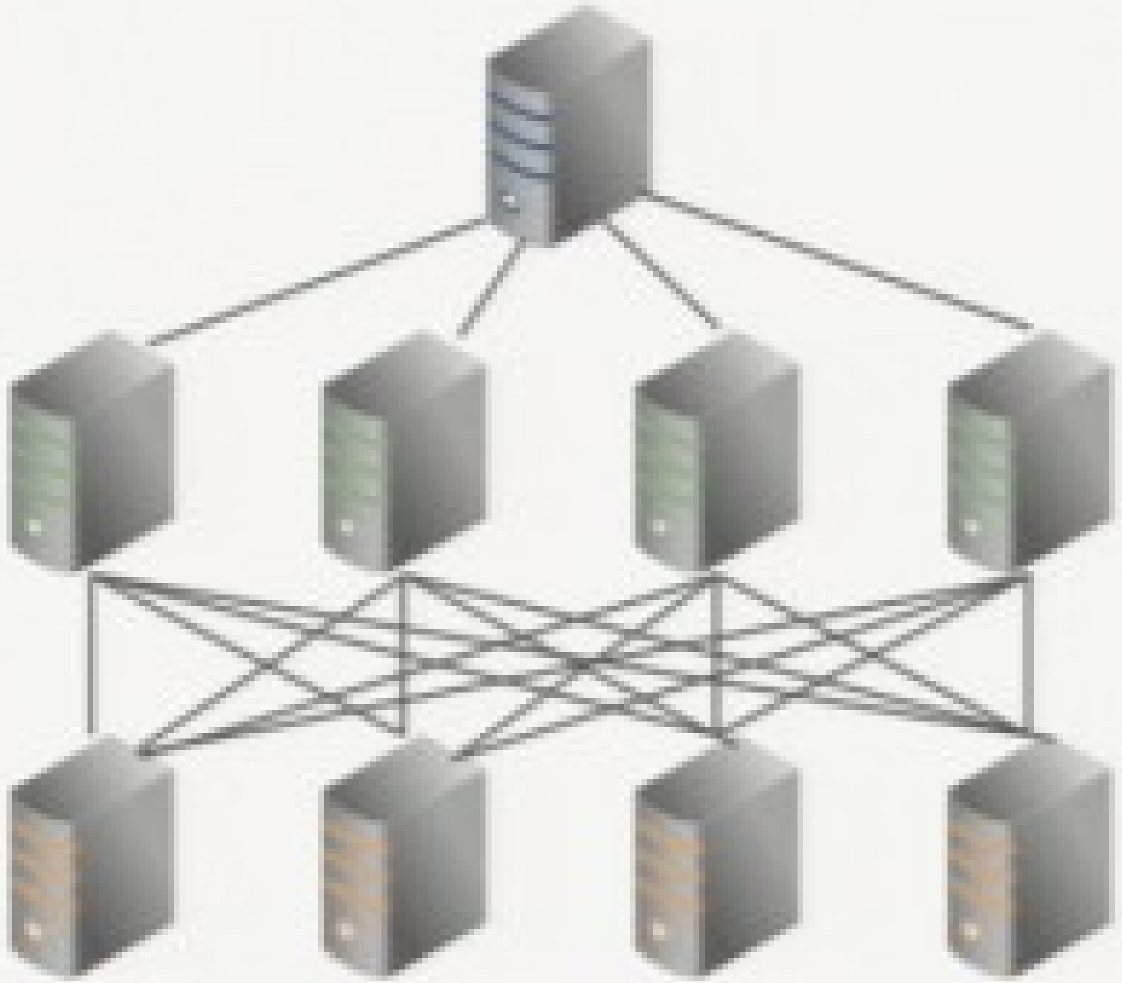
* Active/standby loadbalancing with virtual IP as sinle point of entry

Apache

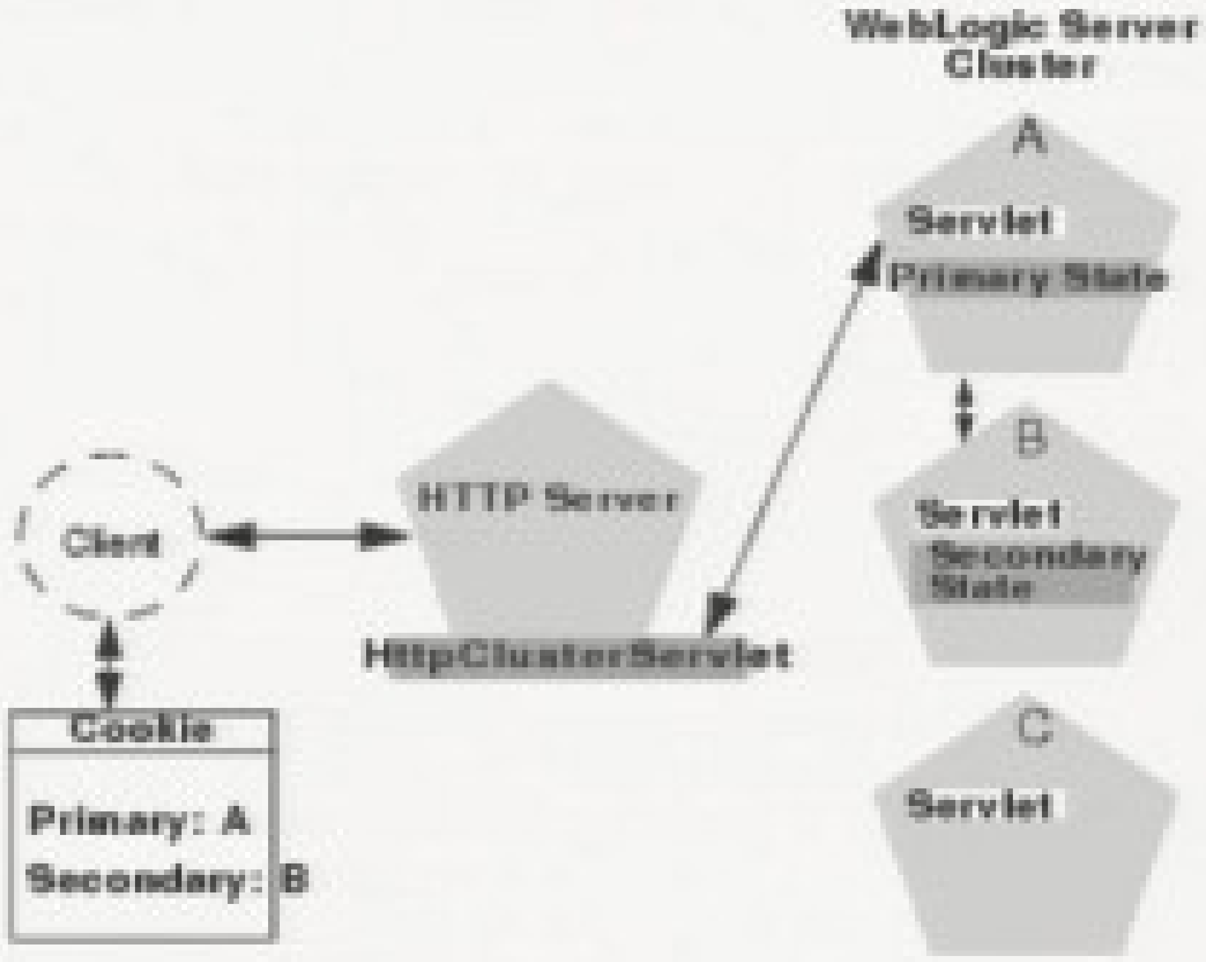
* Terminate SSL encryption
* Serving static content
* Stateless load balancing between available cluster nodes

Weblogic/Tomcat

* Executes application logic



Failover within one datacenter:Apache plugin (mod_wl)

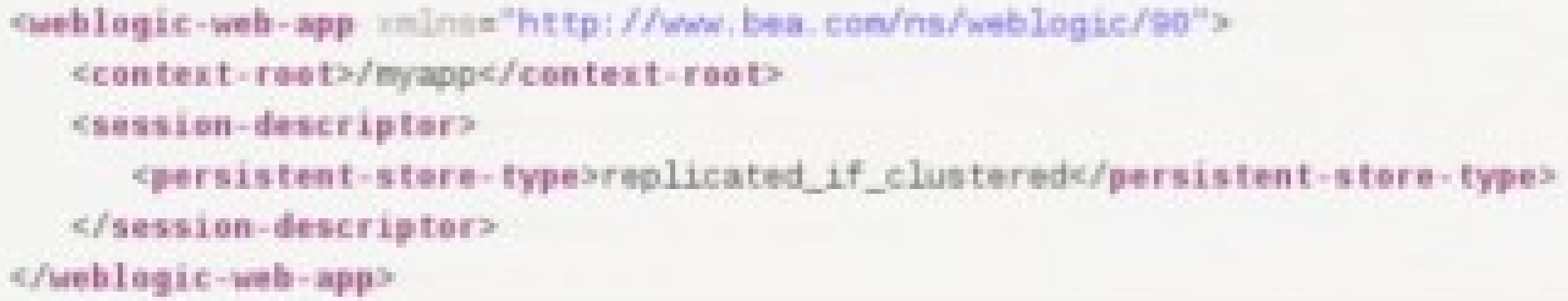


Session ID Format: sessionid!primary_server_id!secondary_server_id

Quelle: http://egeneration.beasys.com/wls/docs100/cluster/wwimages/cluster-06-1-2.gif

Development guidelines (HTTPSession)

- If you need a session then you most probably want to replicate it
- Example (weblogic.xml)



```
<weblogic-web-app xmlns="http://www.bea.com/ns/weblogic/90">
   <context-root>/myapp</context-root>
   <session-descriptor>
      <persistent-store-type>replicated_if_clustered</persistent-store-type>
   </session-descriptor>
</weblogic-web-app>
```

- **Generally all requests of one session go to the same application instance**
  - When it fails (answer with 50x, dies or not answer in a given period) the backup instance is involved
- The session attributes are only replicated on the backup node when **HTTPSession.setAttribute** was called. *HTTPSession.getAttribute("foo") .changeSomething()* will not be replicated!
- Every attribute stored in the HTTPSession must be **serializable**!
- The ServletContext will not be replicated in any cases.
- If you implement **caches** they will have probably different contents on every node (except we use a 3rd party cluster aware cache). Probably the best practice is not to rely that the data is present and **declare the cache transient**
- **Keep the session** small in size and do regular reattaching.

bytesource

Development guidelines (cluster handling)

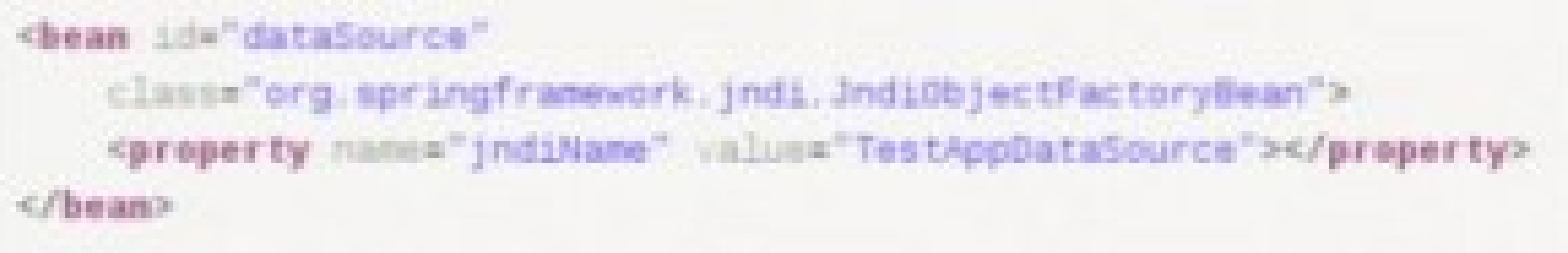
- **Return proper HTTP return codes to the client**
  - Common practice is to return a well formed error page with HTTP code 200
  - It is a good practice if you are sure that the cluster is incapable of recovering from it (example: a missing page will be missing on the other node too)
  - But an exhausted resource (like heap, datasource) could be present on the other node
  - It is hard to implement it, therefore Weblogic offers you help:
  - You can bind the number of execution threads to a datasource capacity
  - **Shut down the node if an OutOfMemoryError occurs** but use it with extreme care!
- **Design for idempotence**
  - Do all your methods idempotent as far as possible.
  - For those that cannot be idempotent (e.g. *sendMoney(Money money, Account account)*) prevent re-execution:
    - By using a ticketing service
    - By declaring the it as not idempotent:

```
<LocationMatch /pathto/yourservlet >
        SetHandler weblogic-handler
        Idempotent OFF
</Location>
```

bytesource

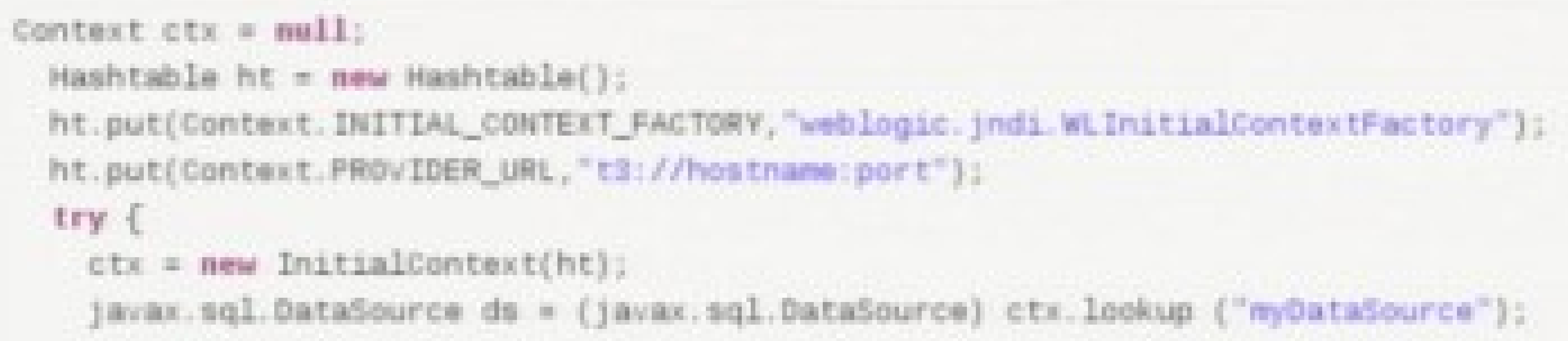
Development guidelines (Datasources)

- **Don't build your own connection pools**, take them from the Application Server by JNDI Lookup
  - As we are using Oracle RAC , the datasource must be a multipool consisting of single datasources per RAC node
    - One can take one of the single datasources out of the mutlipool (online)
    - Load balancing is guaranteed
    - Reconfiguring the pool online
  - Example Spring config:



```
<bean id="dataSource"
    class="org.springframework.jndi.JndiObjectFactoryBean">
    <property name="jndiName" value="TestAppDataSource"></property>
</bean>
```

  - Example without Spring:



```
Context ctx = null;
  Hashtable ht = new Hashtable();
  ht.put(Context.INITIAL_CONTEXT_FACTORY,"weblogic.jndi.WLInitialContextFactory");
  ht.put(Context.PROVIDER_URL,"t3://hostname:port");
  try {
    ctx = new InitialContext(ht);
    javax.sql.DataSource ds = (javax.sql.DataSource) ctx.lookup ("myDataSource");
```

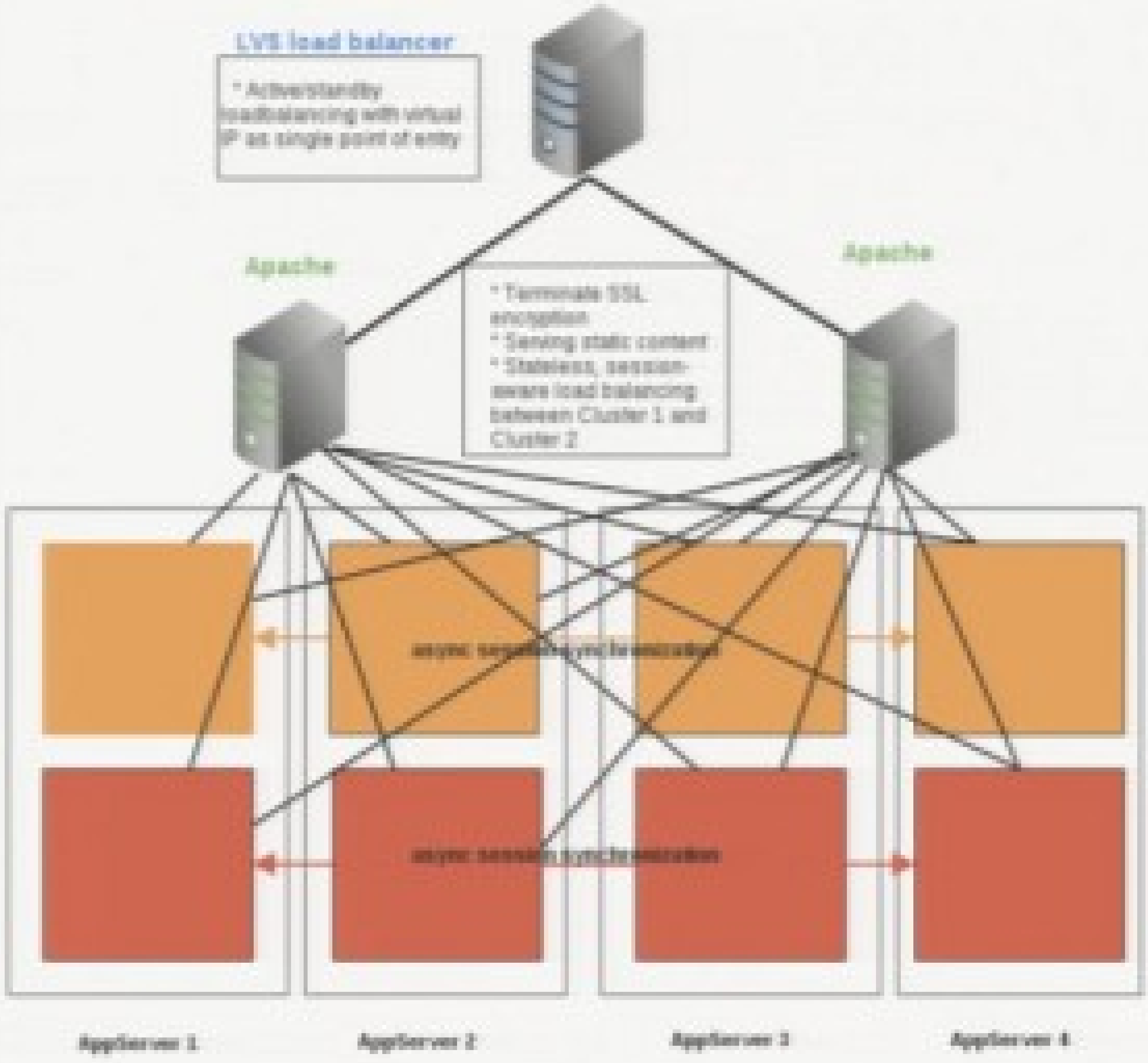
Basic monitoring

- Different possibilities for monitoring on Weblogic
  - Standard admin console
    - Threads (stuck, in use, etc), JVM (heap size, usage etc.), online thread dumps
    - Connection pools statistics
    - Transaction manager statistics
    - Application statistics (per servlet), WorkManager statistics
  - Diagnostic console
    - Online monitoring only
    - All attributes exposed by Weblogic Mbeans can be monitored
    - Demo: diagnostics console
  - Diagnostic images
    - On demand, on shutdown, regularly
    - Useful for problem analysis (especially for after crash analysis)
    - For analysing of resource leaks: Demo: analyse a connection leak and a stuck thread
  - SNMP and diagnostic modules
    - All MBean attributes can be monitored by SNMP
    - Gauge, string, counter monitors, log filters, attribute changes
    - Collected metrics, watches and notifications

bytesource

Zero downtime deployment

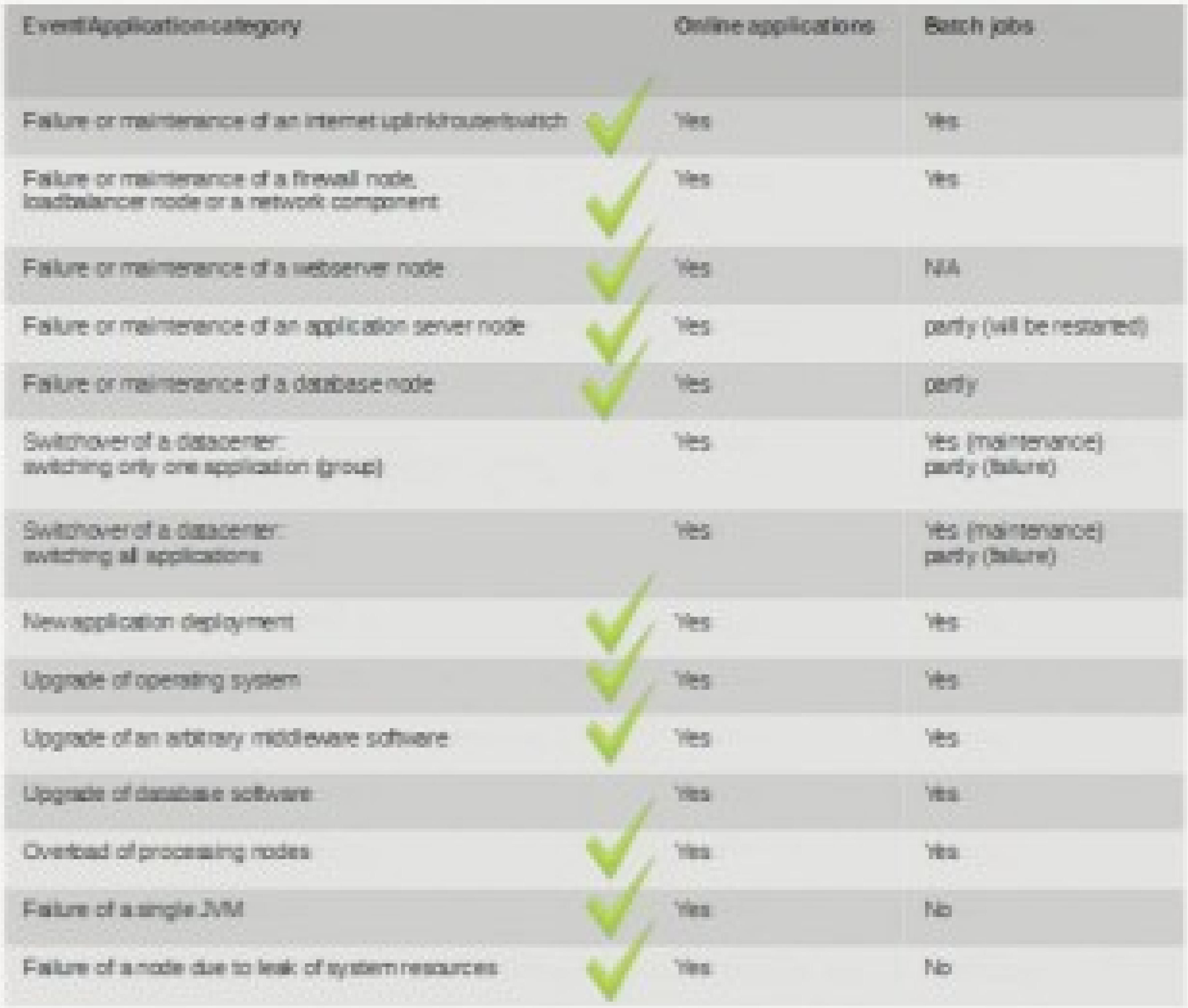
- 2 Clusters within the one datacenter
  - Managed by Apache LB
  - (simple script based on the session ID)
- Both are active during normal operations
- Before we deploy the new release we switch off cluster 1
  - Old sessions go to both cluster 1 and 2
  - New sessions go to cluster 2 only
  - When all sessions of cluster 1 expire we deploy the new version
  - Test it
  - If everything ok, then we put it back
  - into the Apache load balancer
  - Now we take cluster 2 off
  - Untill all sessions expire
  - The same procedure as above
- Then we deploy on the second datacenter


LVS load balancer
* Active/standby loadbalancing with virtual IP as single point of entry
Apache
Apache
* Terminate SSL encryption
* Serving static content
* Stateless, session-aware load balancing between Cluster 1 and Cluster 2
async session synchronization
async session synchronization
AppServer 1
AppServer 2
AppServer 3
AppServer 4

Our Concepts

- ✓ Independent Applications or Application Groups
- ✓ Independent Internet and internal network trafic
- ✓ Reduce/avoid Downtime within a DC
- Replicate the data between the DCs and make the application switchable

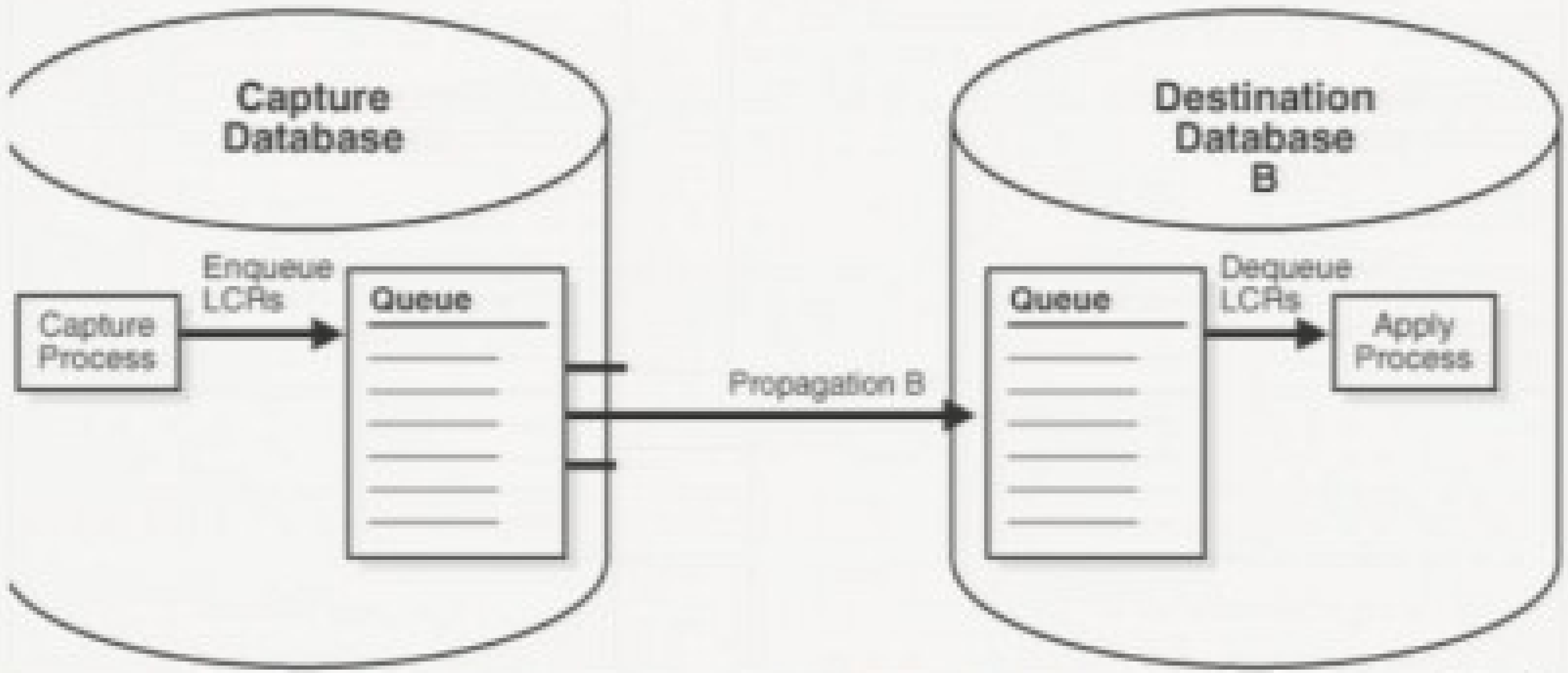
Our requirements again



Event\Application category		Online applications	Batch jobs
Failure or maintenance of an internet uplink/router/switch	✓	Yes	Yes
Failure or maintenance of a firewall node, loadbalancer node or a network component	✓	Yes	Yes
Failure or maintenance of a webserver node	✓	Yes	N/A
Failure or maintenance of an application server node	✓	Yes	partly (will be restarted)
Failure or maintenance of a database node	✓	Yes	partly
Switchover of a datacenter: switching only one application (group)		Yes	Yes (maintenance) partly (failure)
Switchover of a datacenter: switching all applications		Yes	Yes (maintenance) partly (failure)
New application deployment	✓	Yes	Yes
Upgrade of operating system	✓	Yes	Yes
Upgrade of an arbitrary middleware software	✓	Yes	Yes
Upgrade of database software		Yes	Yes
Overload of processing nodes	✓	Yes	Yes
Failure of a single JVM	✓	Yes	No
Failure of a node due to leak of system resources	✓	Yes	No

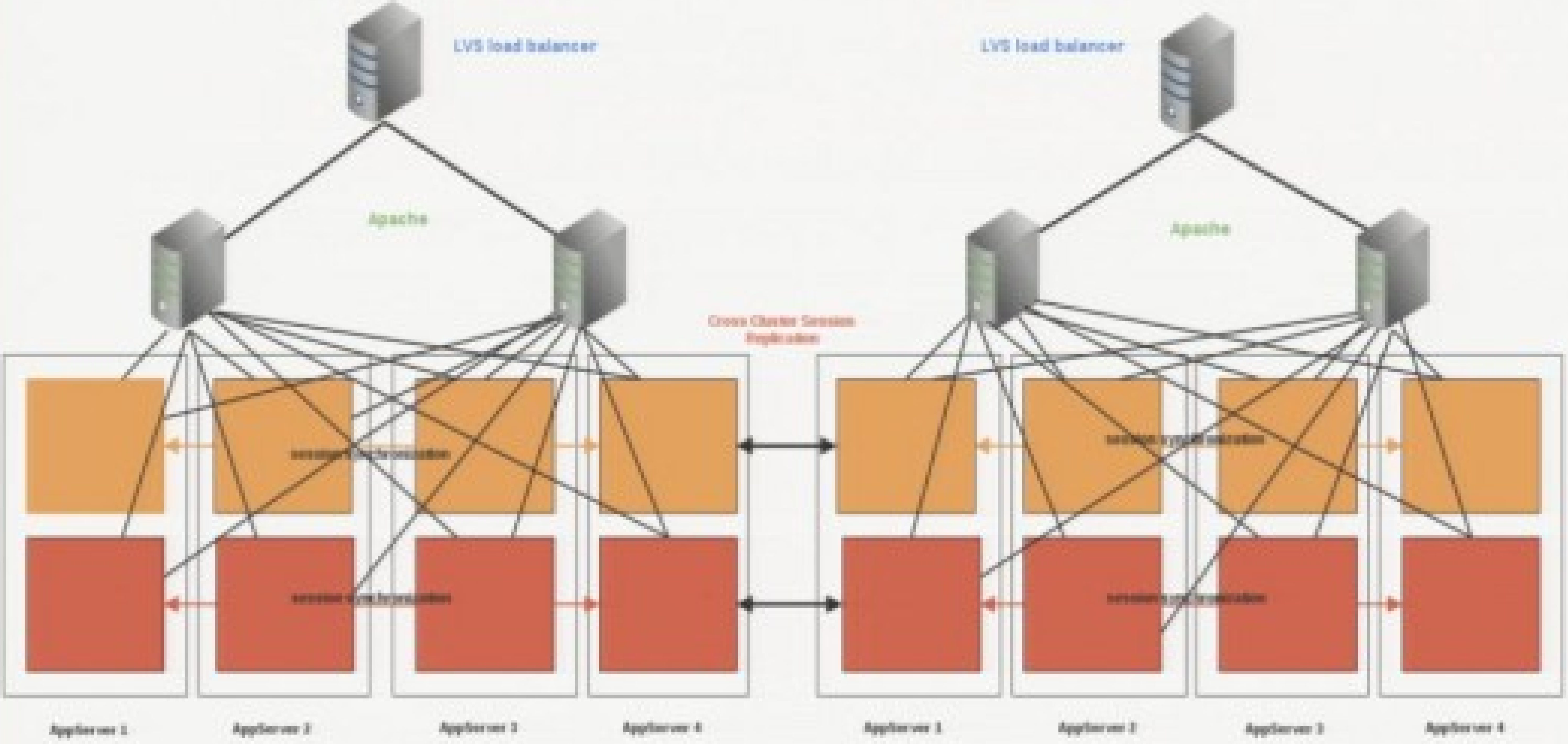
Replicate the data between the DCs

- Bidirectional data replication between DCs
- Oracle Streams/Golden Gate



http://docs.oracle.com/cd/E11882_01/server.112/e10705/man_gen_rep.htm#STREP013

Cross Cluster replication: 2 clusters in 2 datacenters

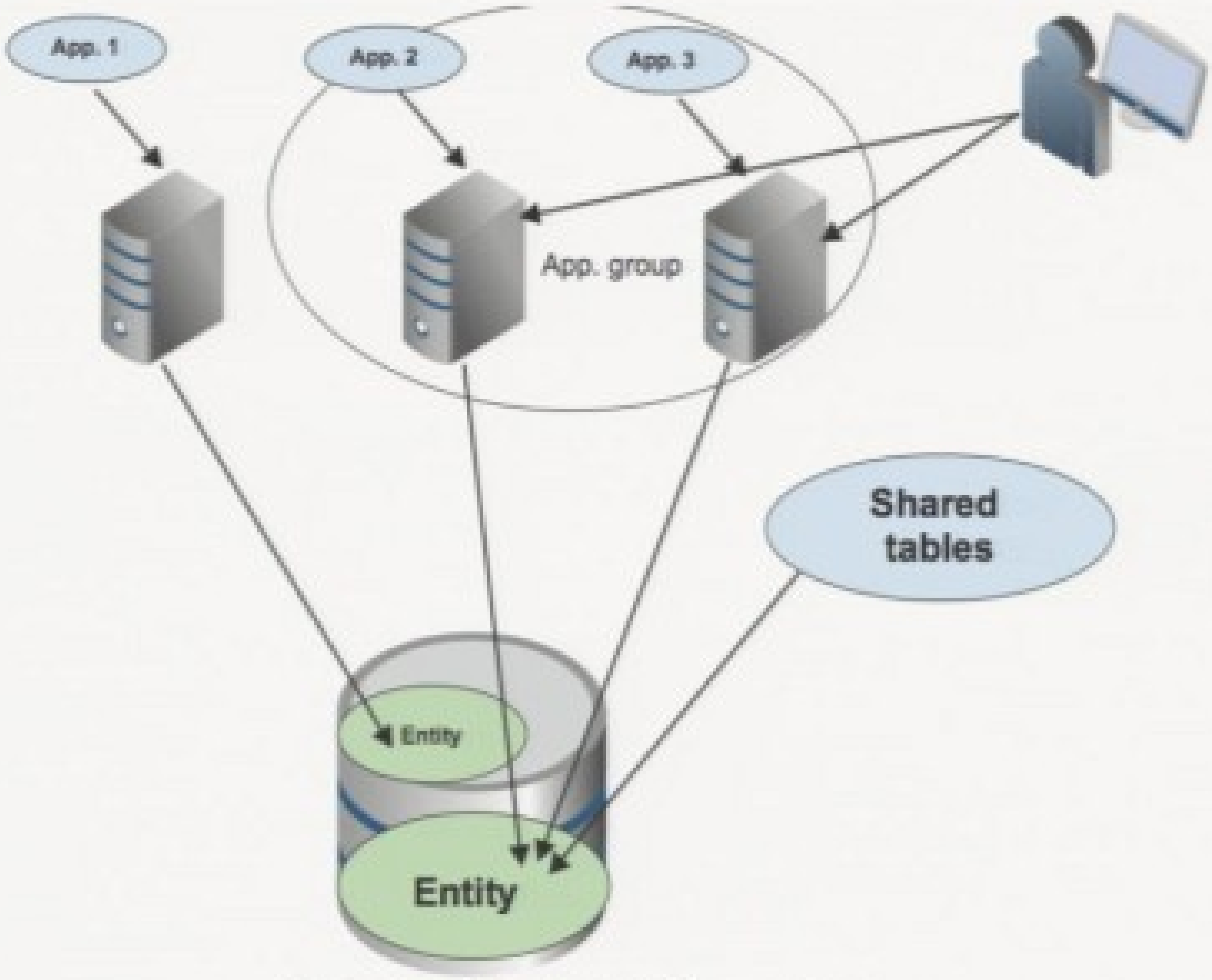


bytesource

Application groups

- One or more applications without hard dependencies to or from other applications
- Why application groups
  - Switching many application at once leads to long downtimes and higher risk
  - Switching a single one is not possible if there are hard dependencies on database level to other applications
  - Identify groups of applications that are critical dependent on each other but not to other applications out of the group
  - Switch such groups always at once
  - As bigger the group as longer the downtime
    - A single application in the category HA will be able to switch without any downtime, just delayed requests
  - Critical (hard) dependencies is if it leads to issues (editing the same record on different DCs will be definitely problematic, reading data for reporting is not)
    - Must be identified on case by case base

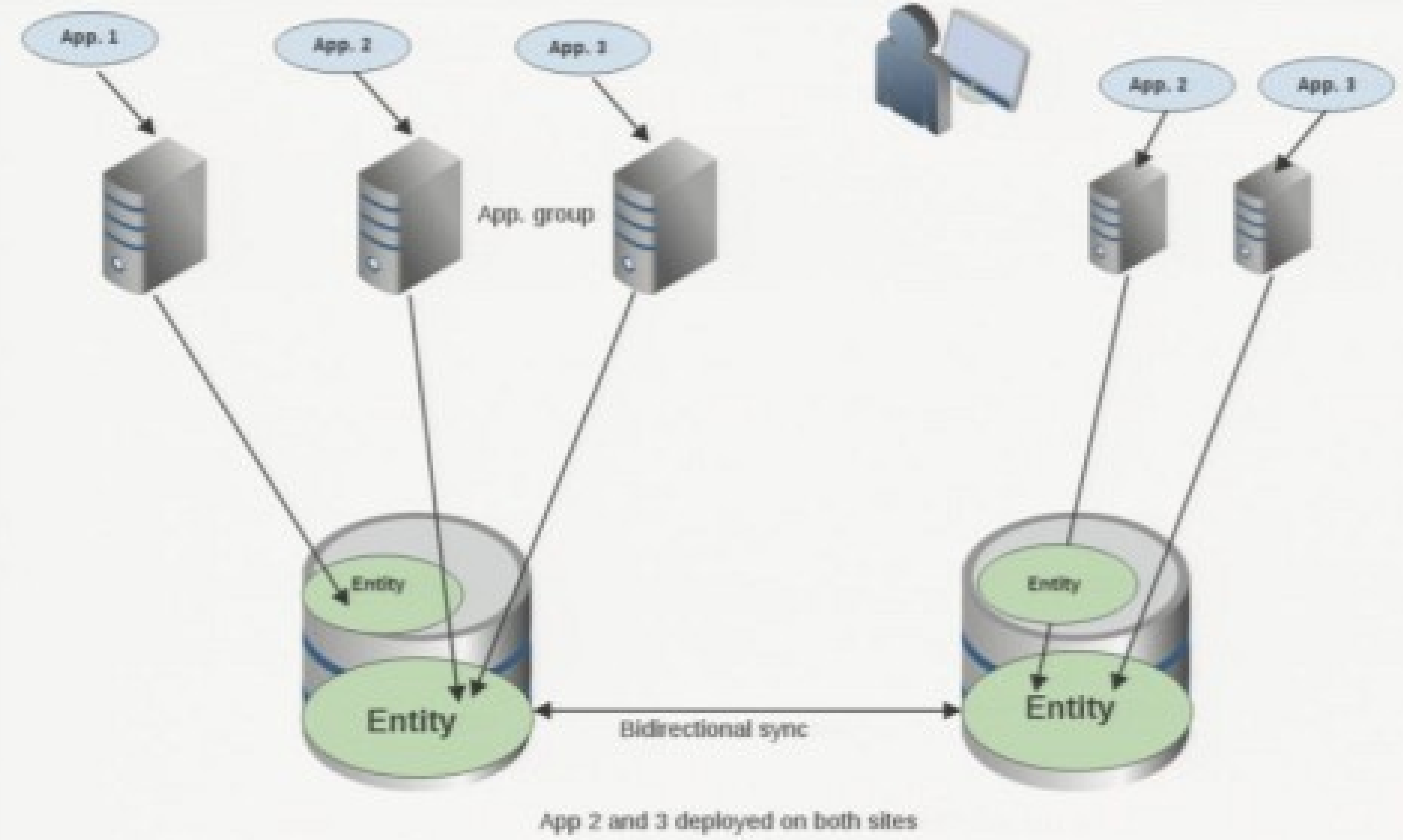
Identify application groups



Before start of migration: identify group of apps

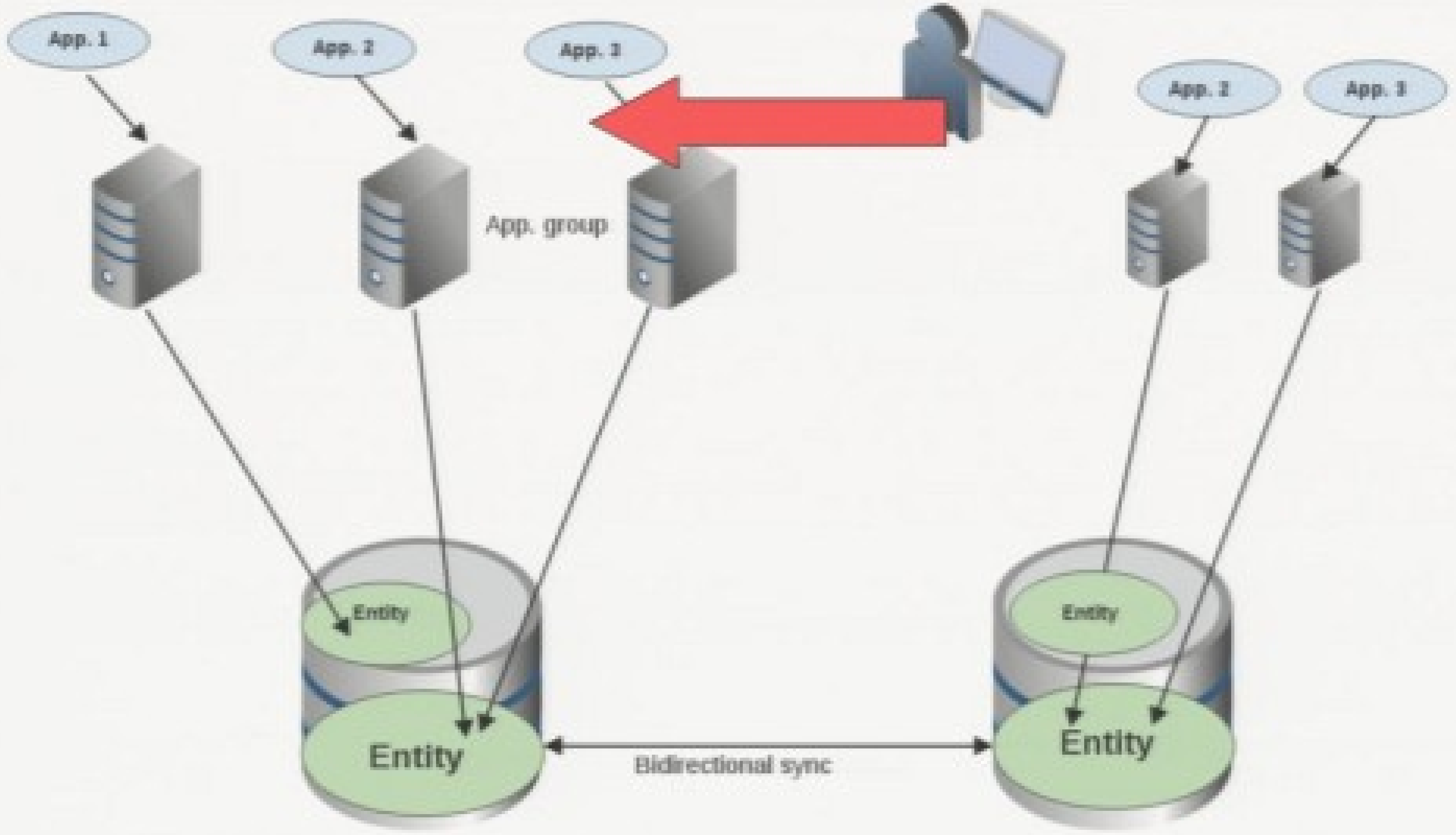
bytesource

Switch application by application



bytesource

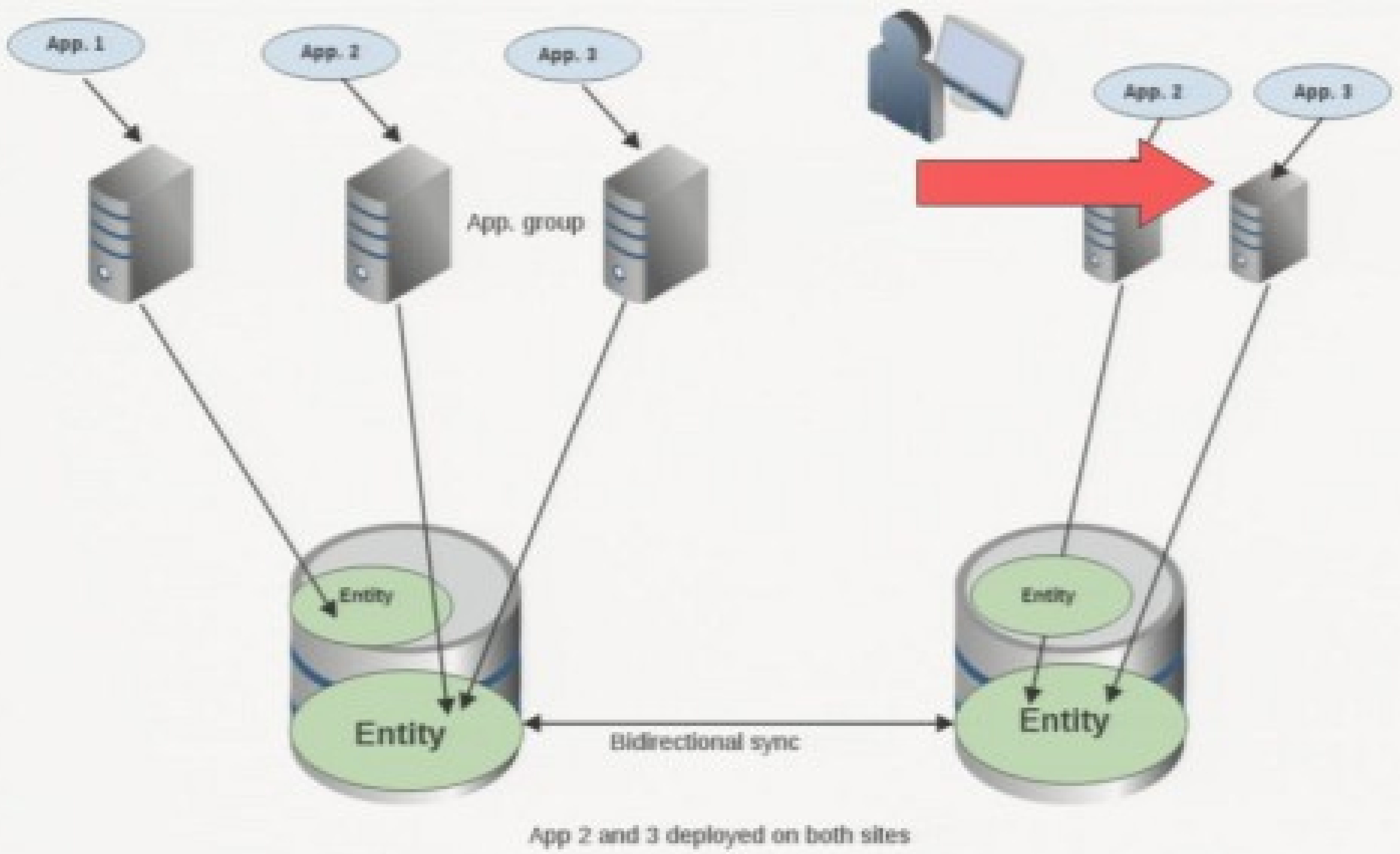
Switch application by application



App 2 and 3 deployed on both sites

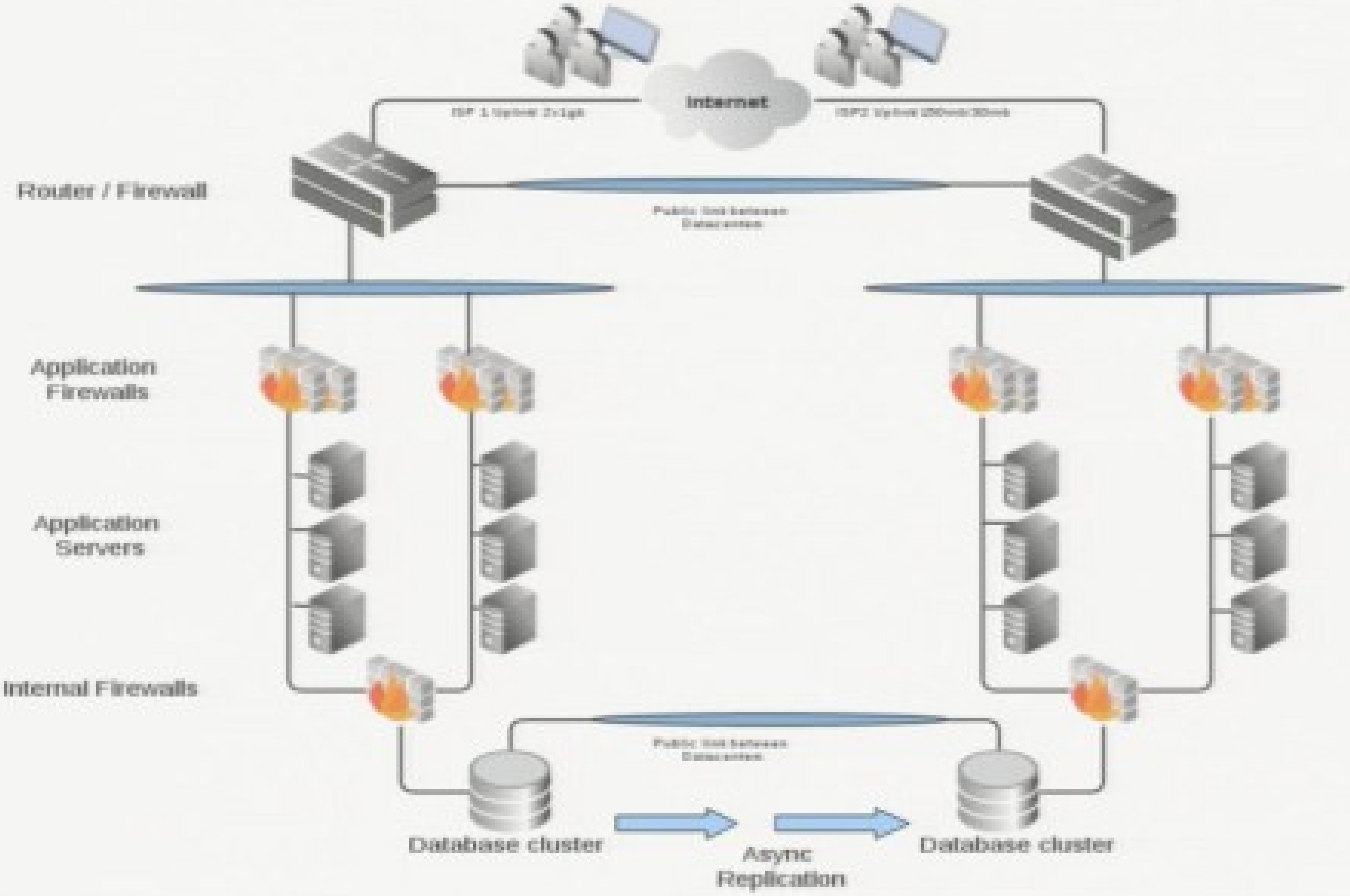
bytesource

Switch application by application

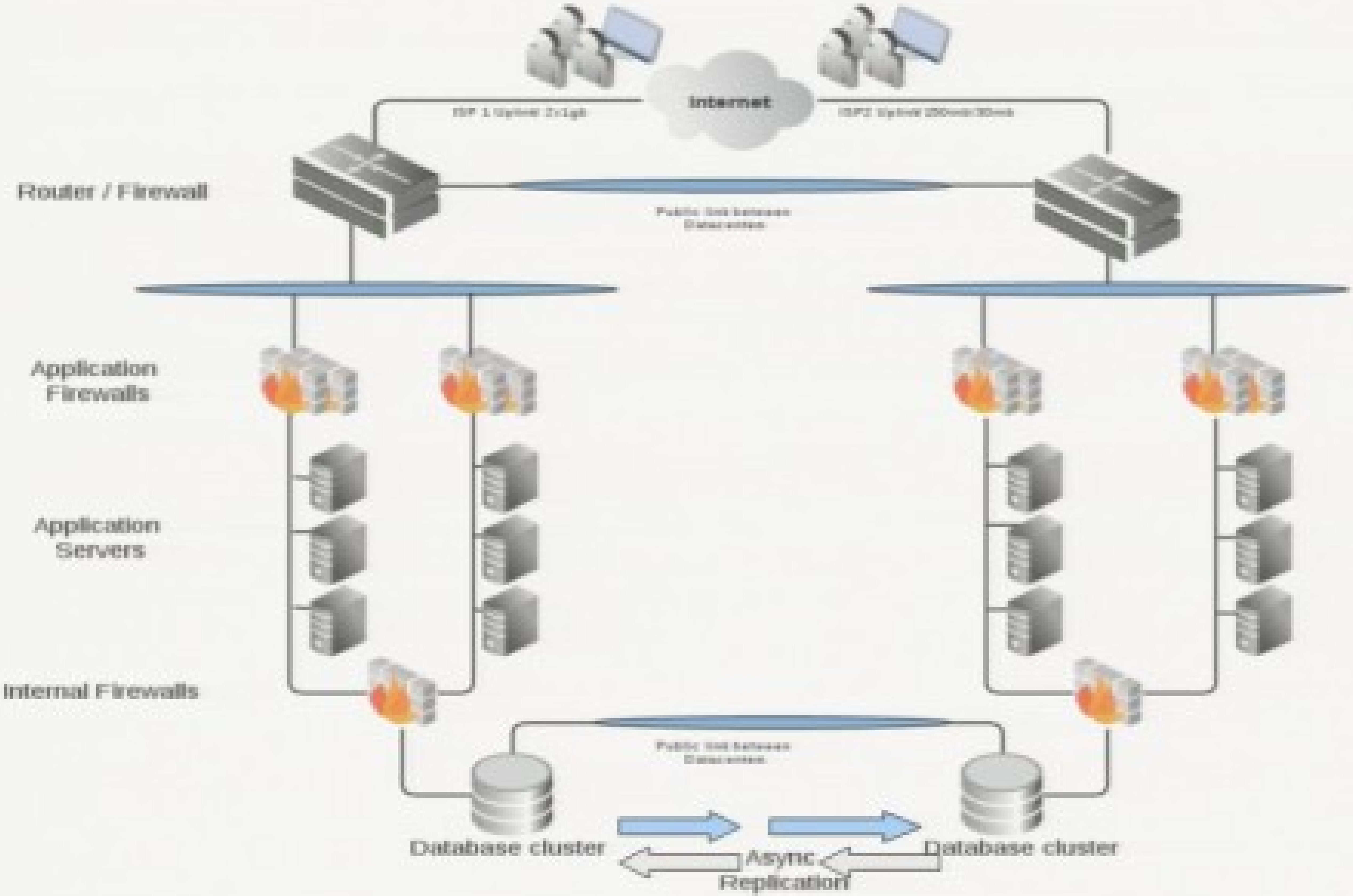


bytesource

Example of a switch procedure of an application group

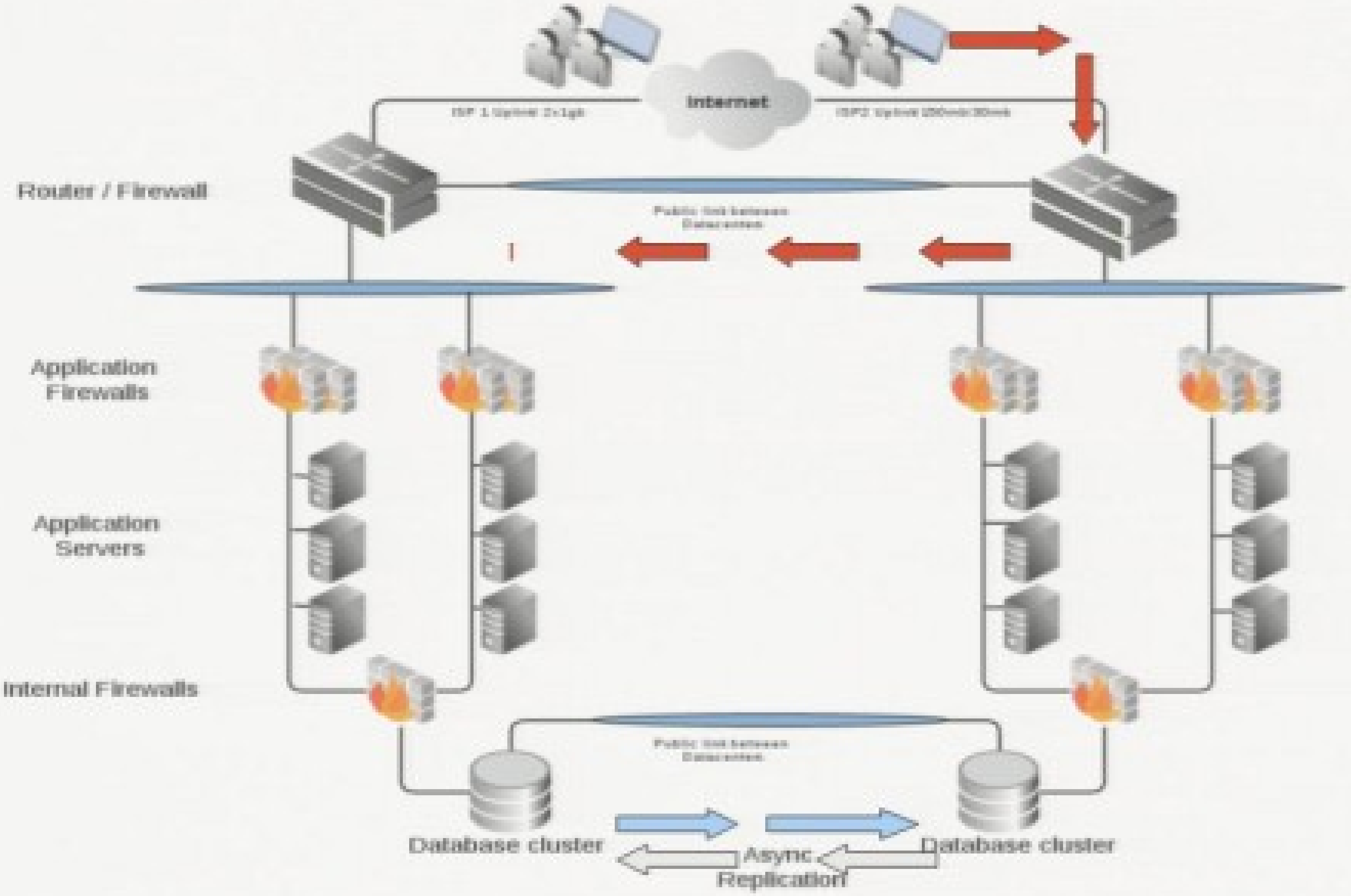


Example of a switch procedure of an application group



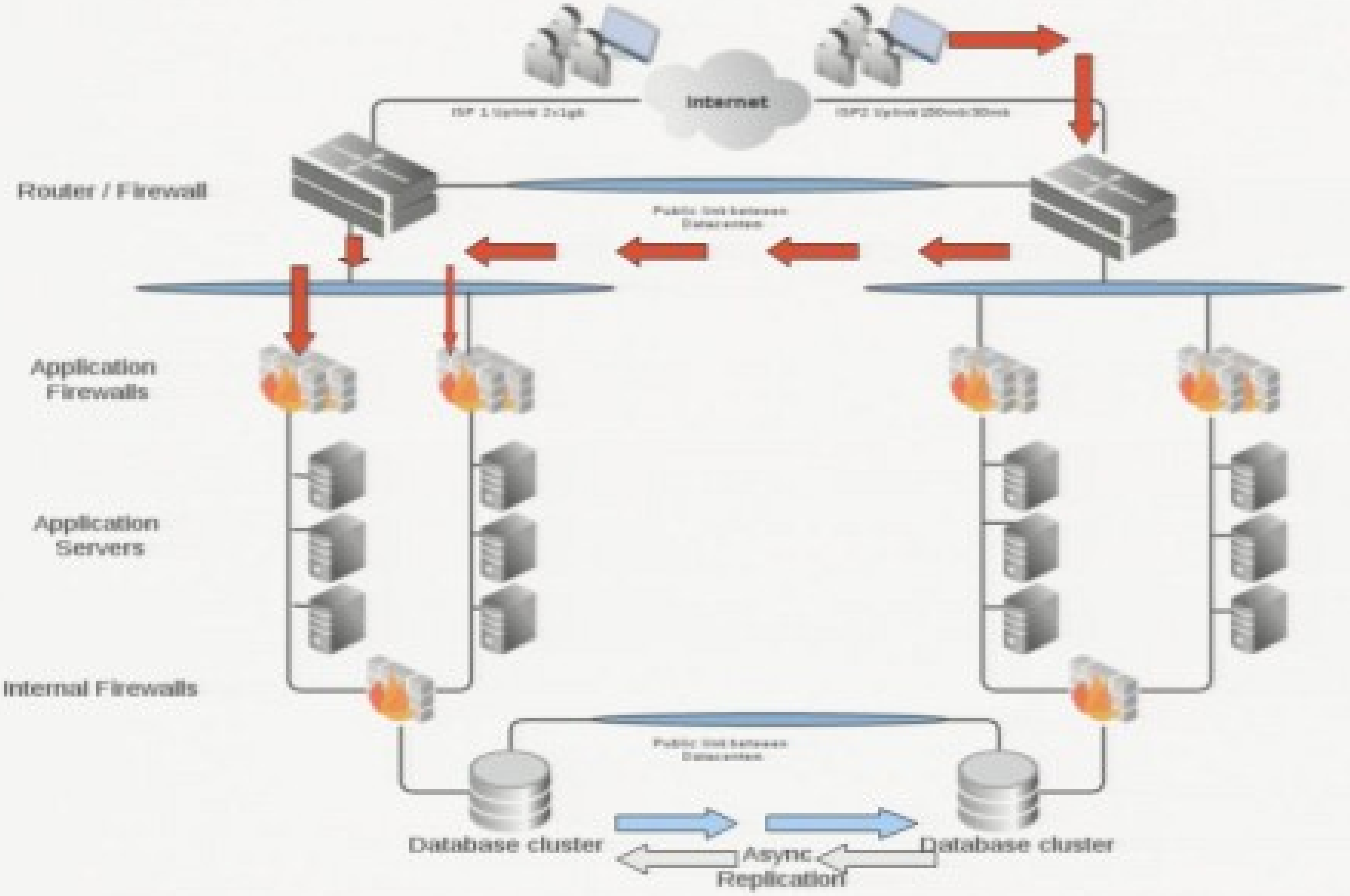
bytesource

Example of a switch procedure of an application group



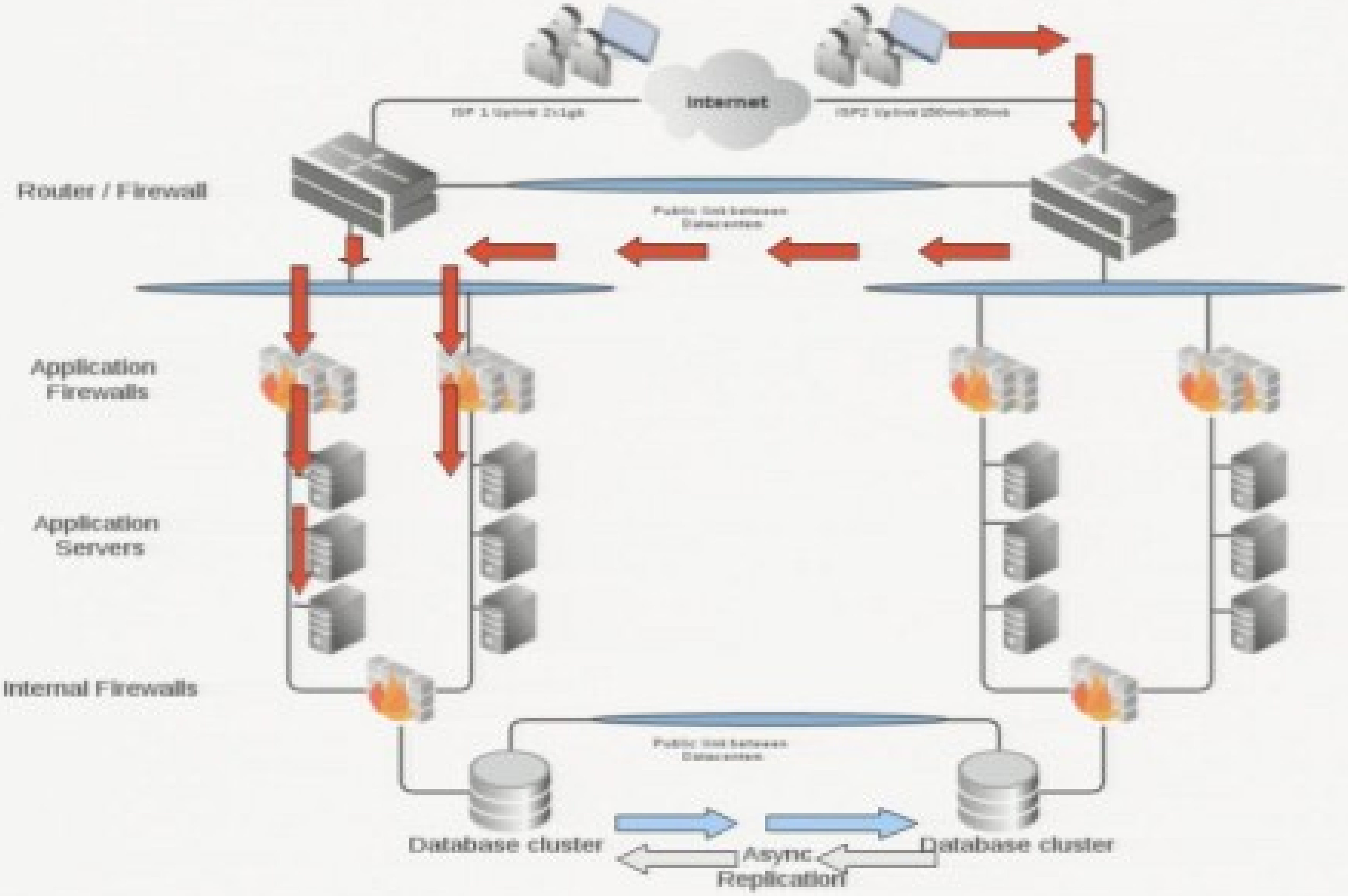
bytesource

Example of a switch procedure of an application group



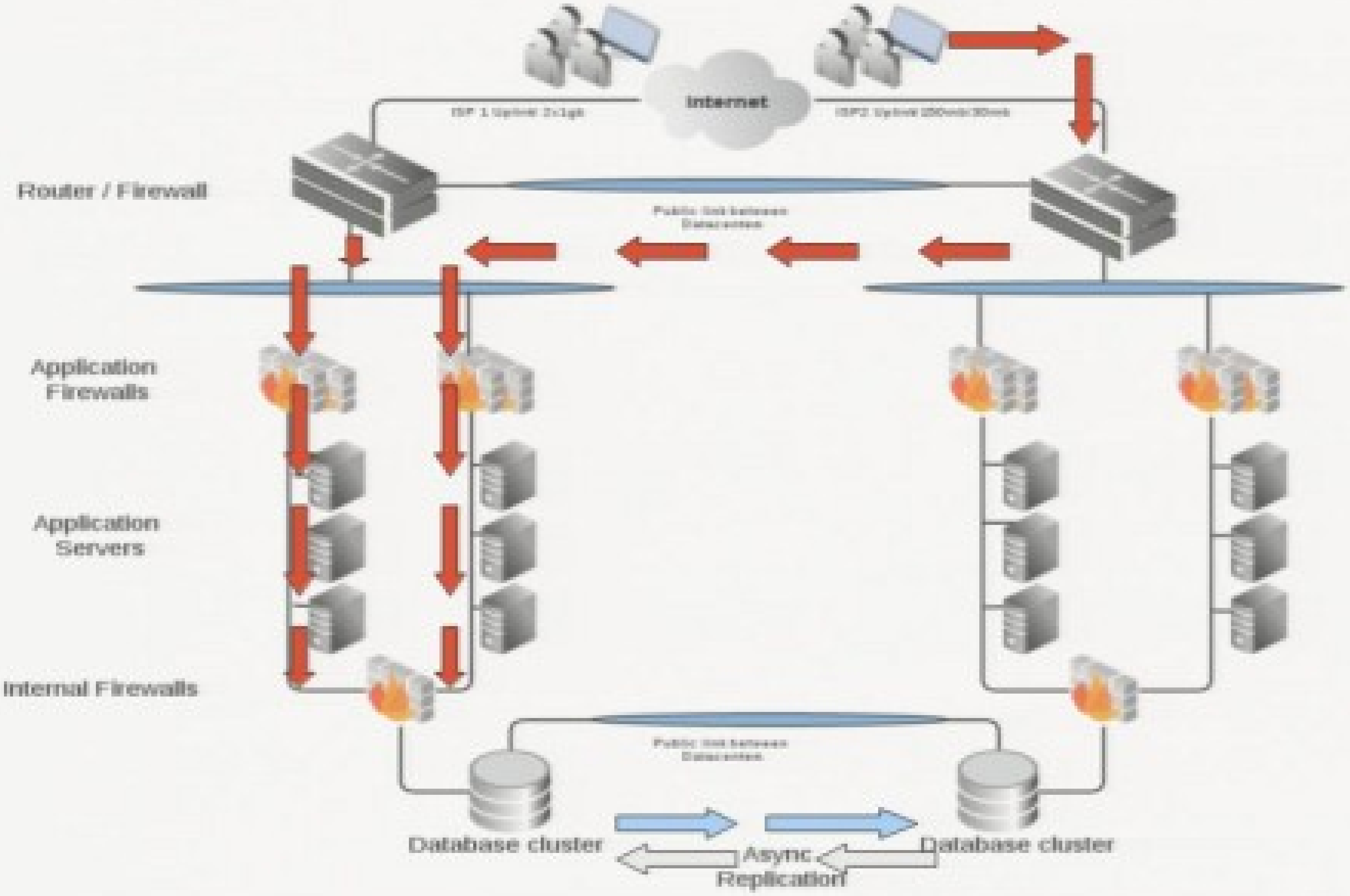
bytesource

Example of a switch procedure of an application group



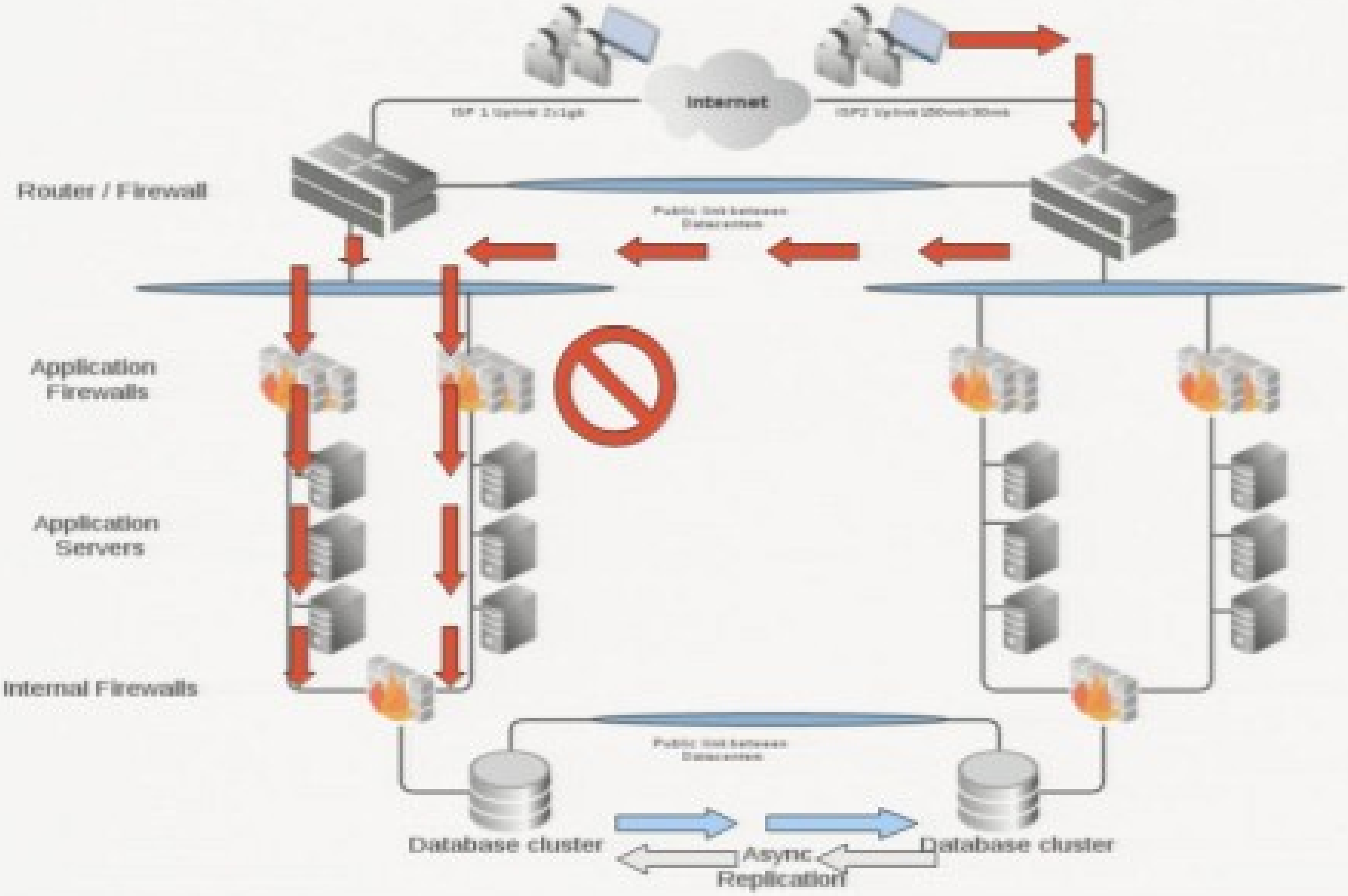
bytesource

Example of a switch procedure of an application group

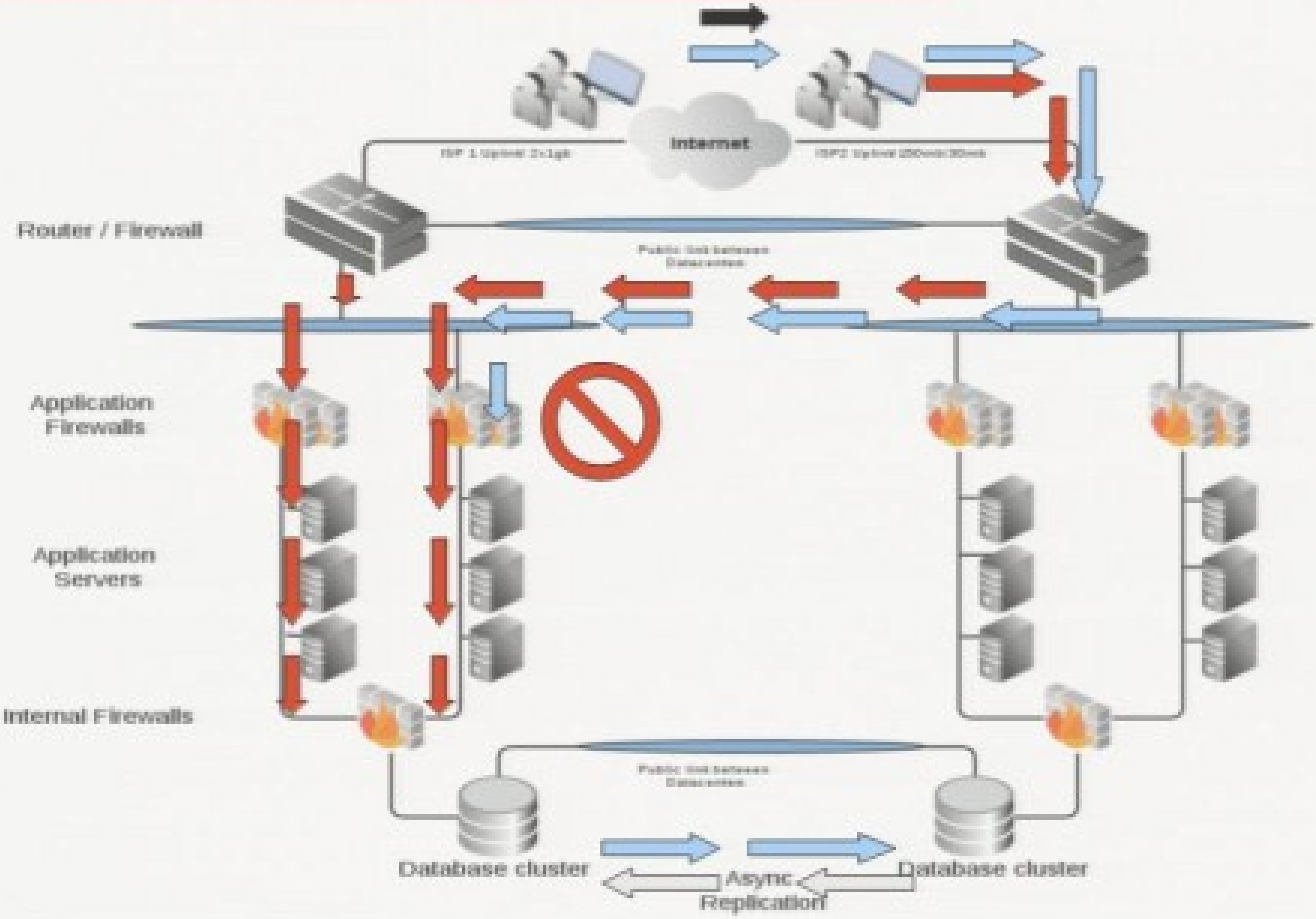


bytesource

Example of a switch procedure of an application group

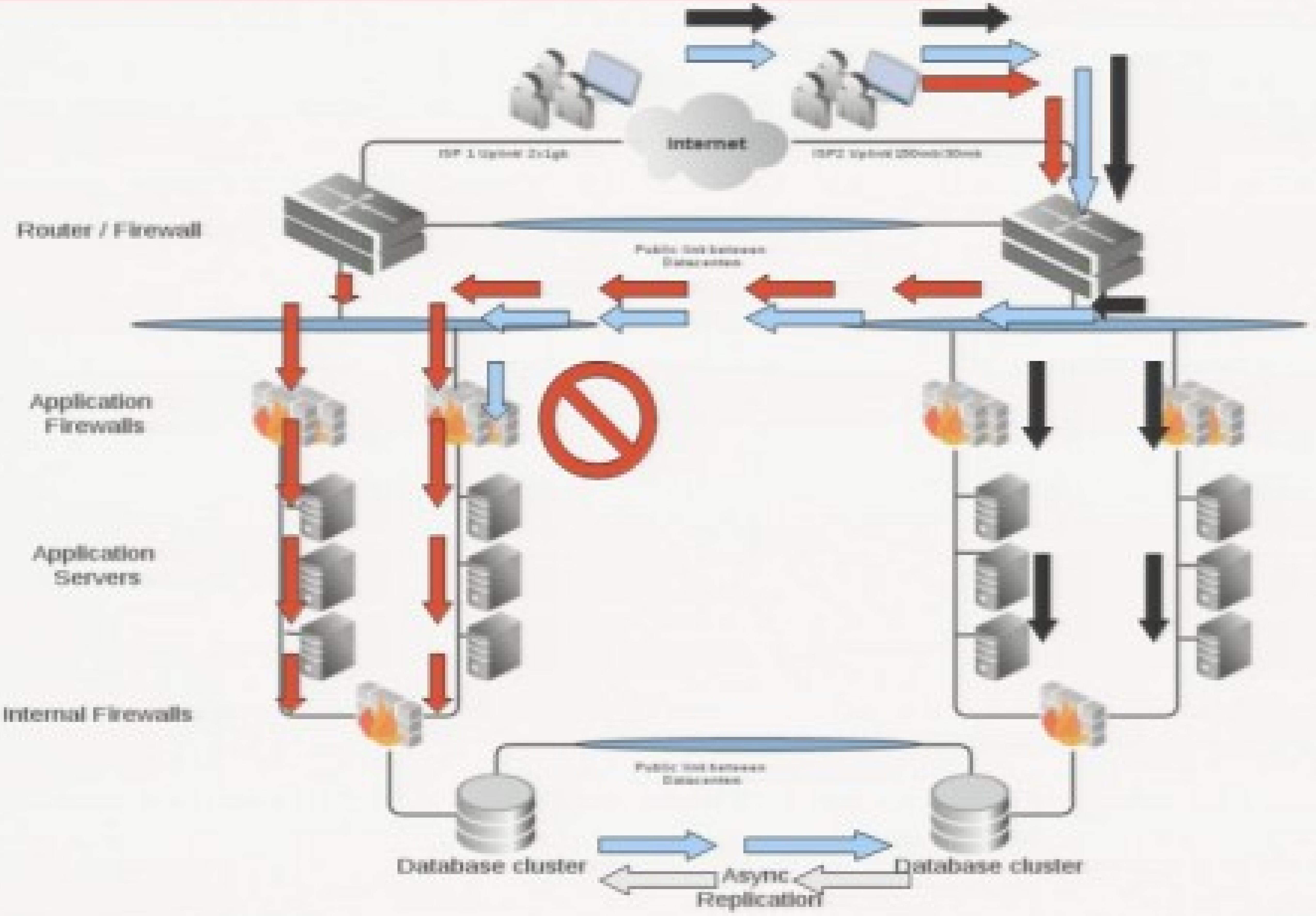


Example of a switch procedure of an application group



bytesource

Example of a switch procedure of an application group



Applications: Limitations

Lim itation/Categories
No bulk transactions
No DB sequences
No file based sequences
No shared file system storage
Use a central batch system
All new releases has to be compatible with the previous release.
Stick to the infrastructure

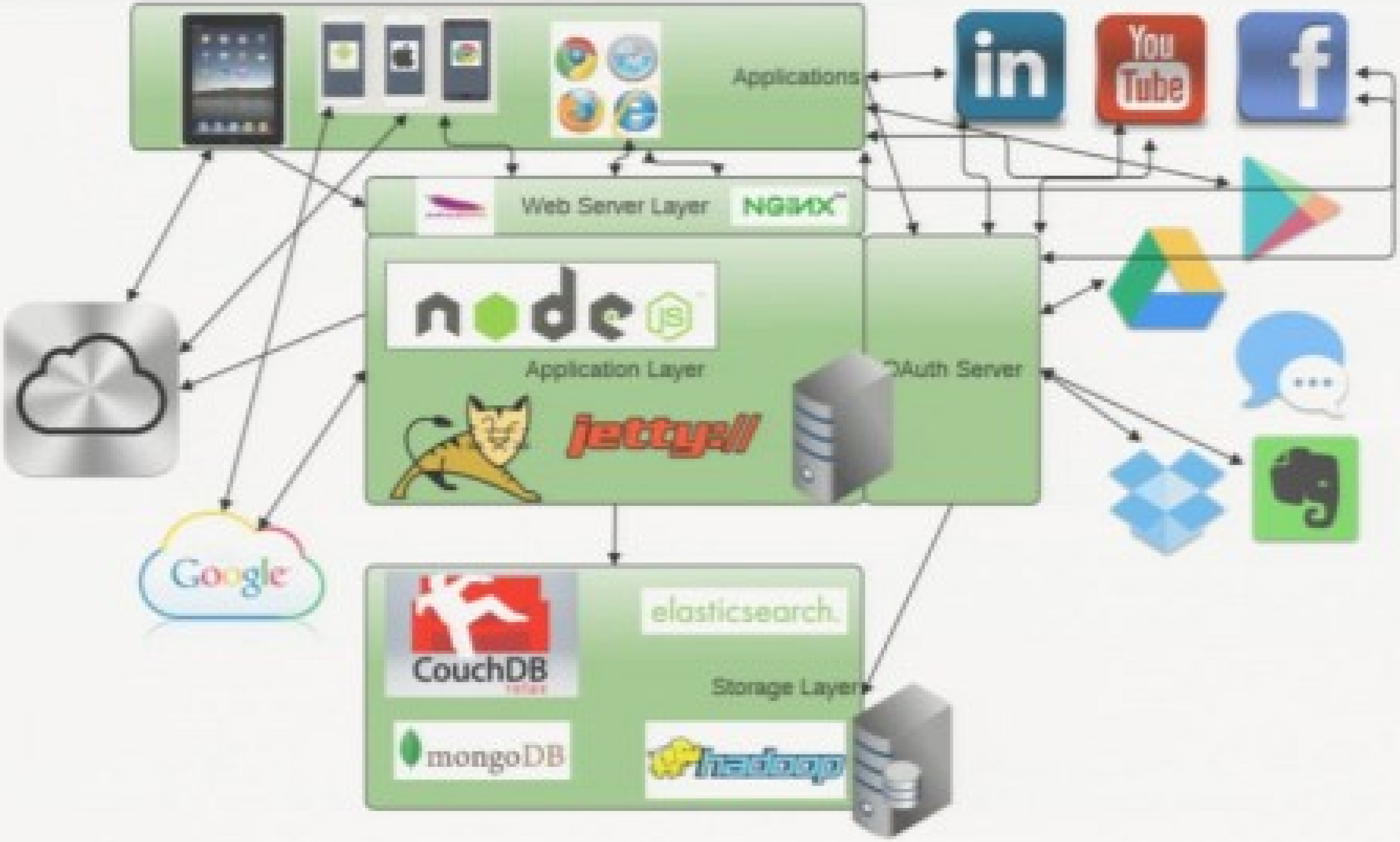
Our Concepts

- ✓ Independent Applications or Application Groups
- ✓ Independent Internet and internal network trafic
- ✓ Reduce/avoid Downtime within a DC
- ✓ Replicate the data between the DCs and make the application switchable

Our requirements once again

Event/Application category		Online applications	Batch jobs
Failure or maintenance of an internet uplink/router/switch	✓	Yes	Yes
Failure or maintenance of a firewall node, loadbalancer node or a network component	✓	Yes	Yes
Failure or maintenance of a webserver node	✓	Yes	N/A
Failure or maintenance of an application server node	✓	Yes	partly (will be restarted)
Failure or maintenance of a database node	✓	Yes	partly
Switchover of a datacenter: switching only one application (group)	✓	Yes	Yes (maintenance) partly (failure)
Switchover of a datacenter: switching all applications	✓	Yes	Yes (maintenance) partly (failure)
New application deployment	✓	Yes	Yes
Upgrade of operating system	✓	Yes	Yes
Upgrade of an arbitrary middleware software	✓	Yes	Yes
Upgrade of database software	✓	Yes	Yes
Overload of processing nodes	✓	Yes	Yes
Failure of a single JVM	✓	Yes	No
Failure of a node due to leak of system resources	✓	Yes	No

Modern Architectures: how does the concepts fit?



bytesource

Modern Architectures: Application Layer

- Web apps
  - Completely independent on the backend
  - Using only Rest APIs
  - 90% of the state is locally managed (supported by frameworks like AngularJS and BackboneJS)
  - Must be compatible with different versions of the Rest API (at least 2 versions)
  - If websockets are used, then more tricky, see backend.
- New mobile versions managed by Apps Stores
  - Good to have a upgrade reminder (to limit the supported versions)
  - Rest API must be versioned and backwards compatible
  - Messages over message clouds is transparent. HA managed by vendors
- Stafeful Services
  - e.g. Oauth v1/v2
    - Normally by DB Persistence

Session Replication

- Less needed that with Server Side Applications
  - Frameworks like AngularJS, BackboneJS , Ember etc. manage their own sessions, routings etc.
- but still needed
  - Weblogic: no change
  - Tomcat evtl. with JDBC Store
  - Jetty with Terracotta
  - **Node.js: secure (digitally signed) sessions stored in cookies**
    - Senchalabs Connect
    - Mozilla/node-client-sessions
      - https://hacks.mozilla.org/2012/12/using-secure-client-side-sessions-to-build-simple-and-scalable-node-js-applications-a-node-js-holiday-season-part-3/

Backend: Bidirectional Data Replication

- Elastic Search ⊖
  - Currently no cross cluster replication
  - But is on their roadmap
- Couchdb ✓
  - Very flexible replication, regardless within one or more datacenters
  - Bidirectional replication is possible
- Mongodb ❗
  - One direction replication possible and mature
  - Bidirectional not possible in the moment
  - Workaround would be: one mongodb per app and strict separation of the apps
- Hadoop HDFS ⊖
  - Currently no cross cluster replication available
  - e.g. Facebook wrote their own replication for HIVE
  - Will possibly arrive soon with Apache Falcon http://falcon.incubator.apache.org/

Questions?

Thank you for your attention !