Advanced Registry Operations Curriculum

Advancd Registry Architectures

Robust, Reliable, and Resilient Registry Operations

Registry Definitions

What's a (ccTLD) registry ?

- Publishes one or more zones (think TLD and SLD)
- Manages delegations
- Publishes public (!) information (WHOIS)
- Possibly, receives payment for the service

Data flows

Inputs

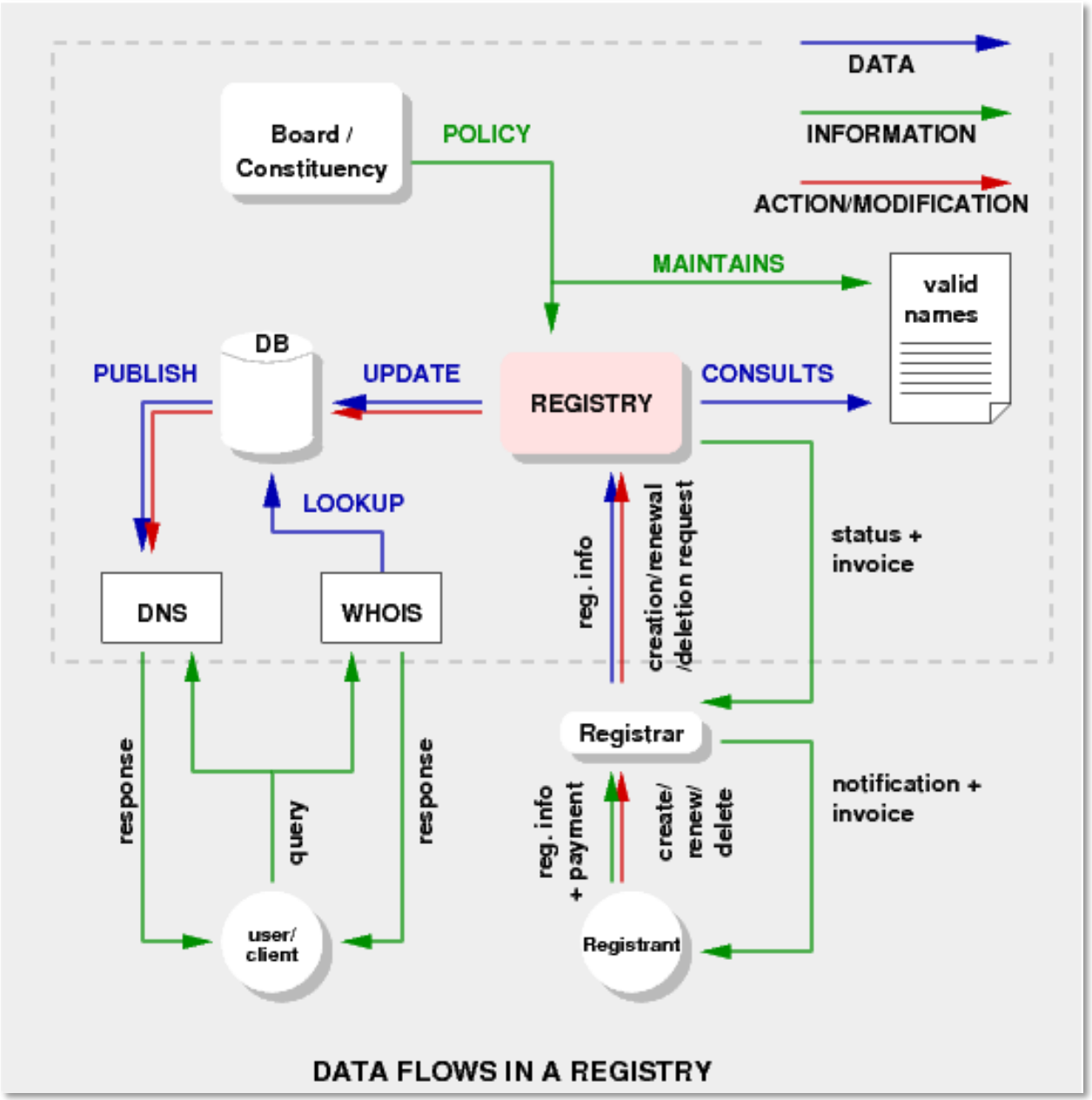
- Domain Name System requests
- Creation/deletion/modification requests of domain names
- Add nameservers (name + IP)
- Administrative information (registrant, tech contact, billing contact, ...)

Data flows

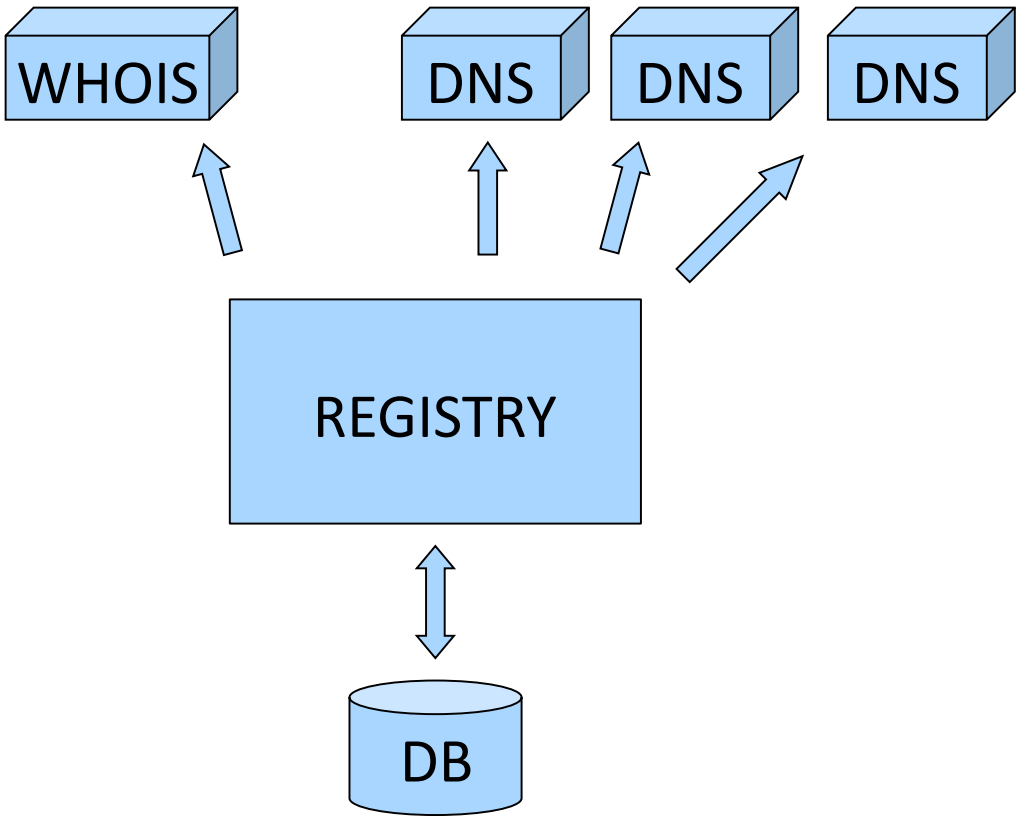
Output:

- Answer DNS queries
- Zones with delegations (publication)
- Glue records (for nameservers which are within the zone being delegated)
- Publication of WHOIS

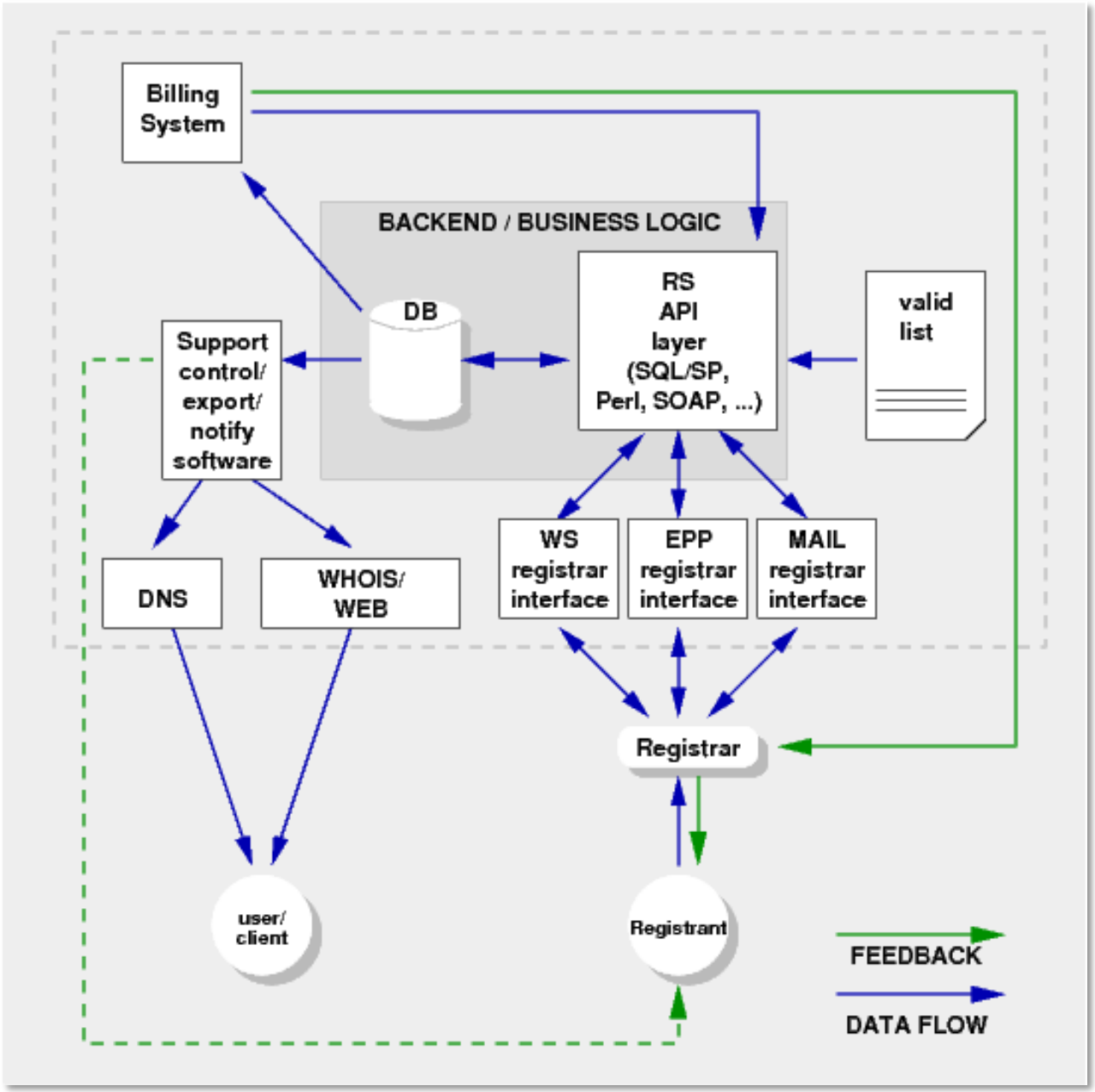
Registry flows



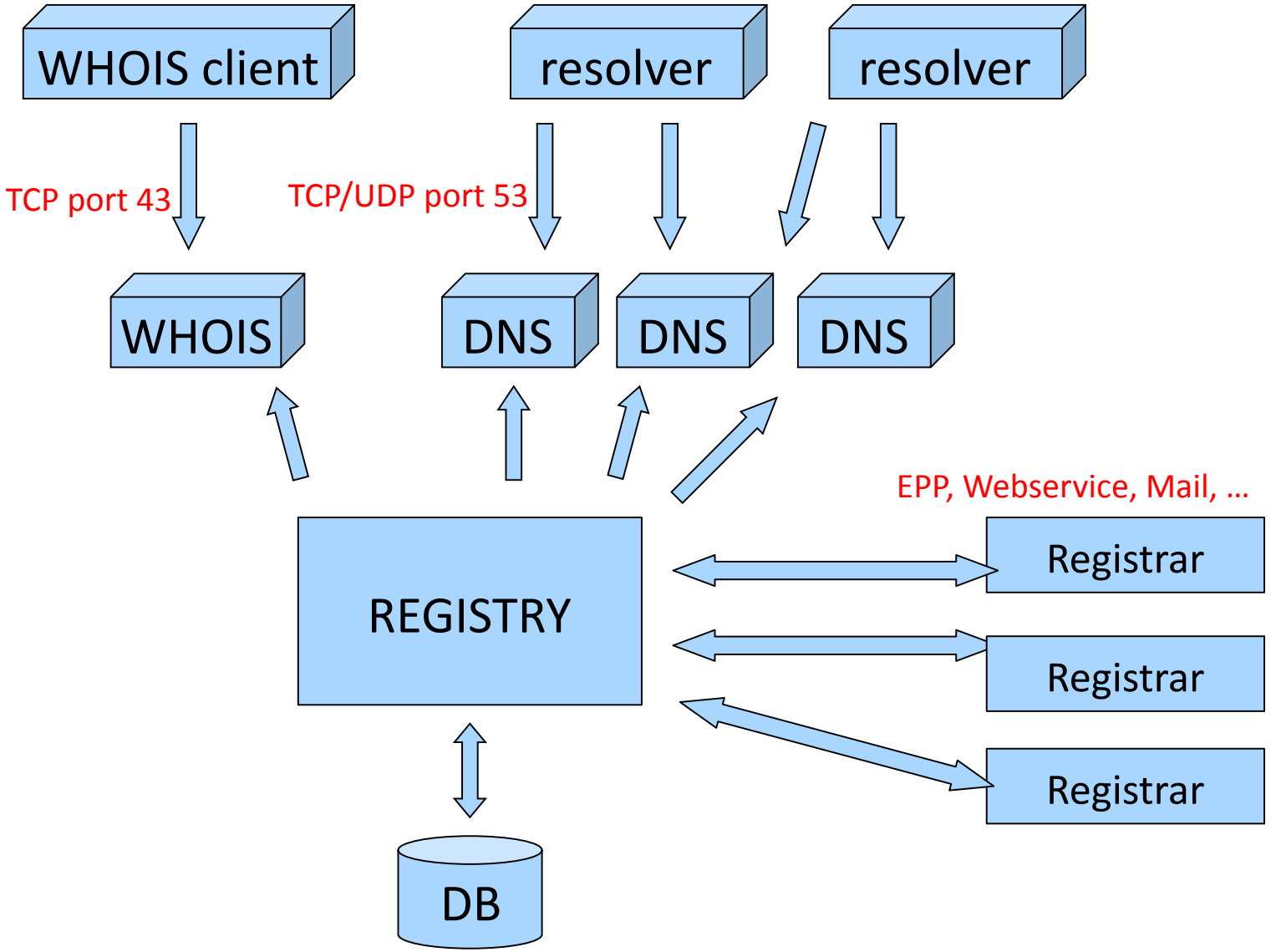
Architecture



Registry architecture



External interfaces



What operations ?

- Add and remove records (redelegation)
- Add/modify/remove nameservers (modification is effectively a redelegation)
- Update of administrative data (whois info)

How complex can it get ?

- It can be as simple as a text zone file with comments in it
- Maintained with Ten Finger Interface

```
...
; SomeCompany
; contact John Dough, +1 123 123 4567, ;
  john@somecompany.mytld
somecompany      NS       ns1.othertld.org.
                 NS       ns.somecompany
ns.somecompany             A        1.2.3.4
...
```

Pretty simple operational model

- Add a delegation
  - Creation of domain
- Change a delegation
  - Domain dedelegation
- Remove a delegation
  - Domain destruction
- Every operation can impact delegation entries, glue records, whois data

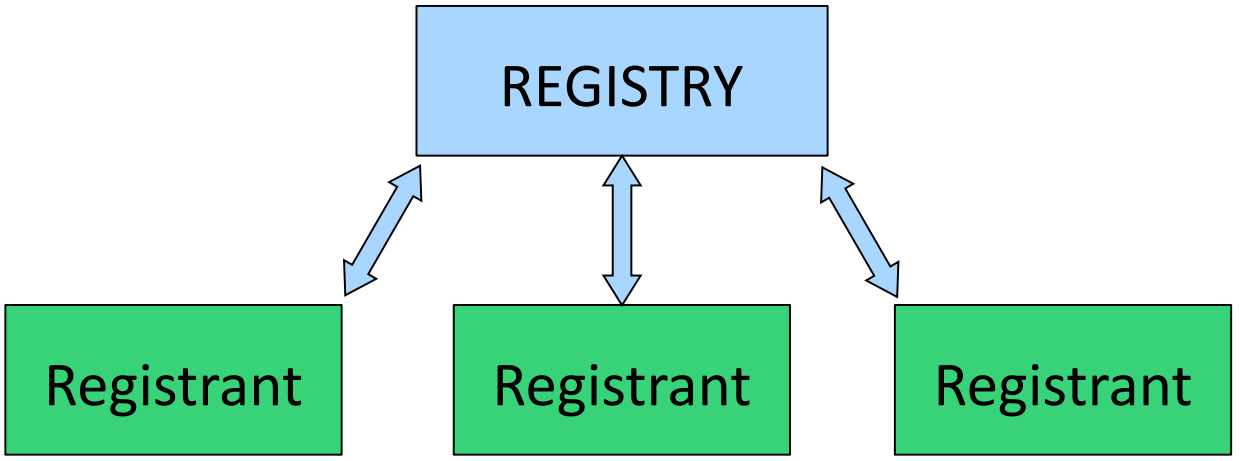
Terminology

- **Registry:** Institution or organisation which maintains the zone and administrative data
- **Registrant:** Physical or moral person which is responsible for a domain name
- **Registrar:** Organisation managing domain registrations on behalf of registrants

Different models: 2R

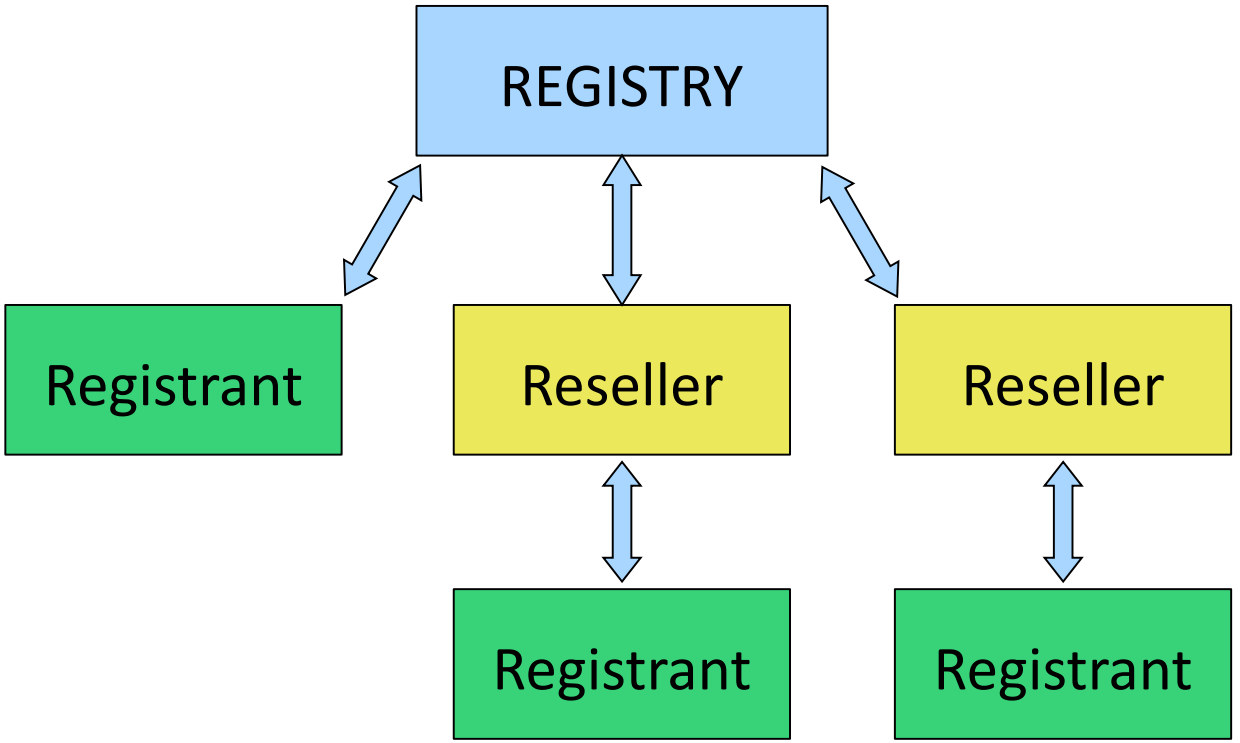
Simple registry models – no registrars

The registrant is in direct contact with the registry. This is also called a ”single access” registry.



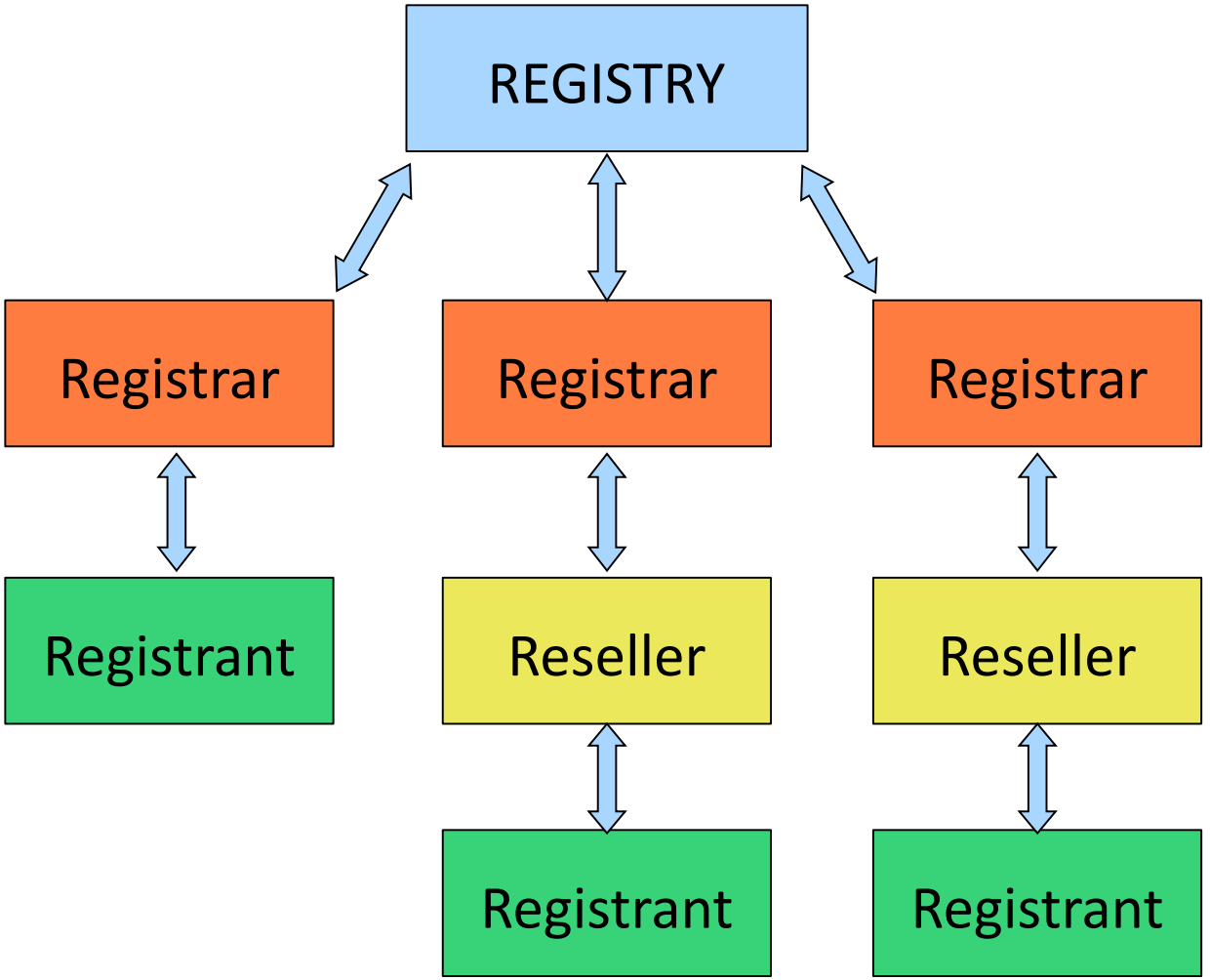
Different models: 2R

It remains a single access registry, even though it may or may not allow resellers:



Different models: 3R

Shared access registry



Thick vs. Thin

- Indicates how the WHOIS is placed/ distributed
- Dependes on where the DB is located
  - Thin: .COM, .NET: administrative data are spread across the various registrars
  - Thick: .INFO – the administrative data are centralized at the Registry

Flat or hierarchical

- Flat
  - A flat design allows any name directly under the top-level country-code domain (i.e., the second-level domain or SLD). For example, nsrc.cctld .
- Hierachical
  - A hierarchical design provides categorized or affinity groups at the second-level. For example, mycollege.edu.cctld, where "edu" specifies educational institutions.

Evolution of a registry

From most simple…

- Text zone file with comments
- Domain registration via email
- No whois, or manually updated
- No registrars or resellers, 2R

…To most complex

- Relational database, Transaction, automated billing
- WHOIS, EPP, Web interface
- 3R with multiple registrars
- Anycasting of DNS servers

EPP

- RFC3730
- Supercedes RRP (RFC2832)
- Extensible Provisioning Protocol
- Based on XML
- Used by an increasing number of registries and registrars
- Not all ”modern” registries have adopted it yet!
- RFC4310 describes the DNS security Extension Mapping for the EPP

WHOIS

- Fetch meta-information about a domain, including administrative data (name, address, phone contact, …)
- RFC 954
  - Not formally specified as a protocol
  - Output from different Registrars and Registries can look different (and often does)
- RFC 3912
  - TCP port 43

Registry-Registrar relationship

Registry-registrar relationship: the accreditation

- Usually, the relationship between a registry and a registrar is based on a contract.
- Some registries require that companies applying to become a registrar to follow an accreditation procedure.
- Criteria to be accredited:
  - Technical stability
  - Corporate status
  - Financial stability
  - Organizational stability
  - Other

Registry-registrar relationship: some contractual aspects

When the relationship between the registry and the registrars is based on a contract, the registry should take into account some aspects:

- Contract transfer and related domain names transfer
- Rescue procedures for those registrants (and domain names) whose registrars "disappeared" or went bankrupt
- Penalties in case the registrar is not up-to-date with the payments to the registry

Registry-registrar relationship: the code of conduct

- **Code of conduct:**
  - In order to ensure that the domain name holder can count on reliable information and a quality service, some registries have proposed a code of conduct to registrars

  - Most code of conducts are based on voluntary principles, but **help the users trust the overall process**

Registry-registrar relationship: communication methods

Communication tools:

- E-mail lists
- Regular meetings
- Help-desks
- Newsletter
- Dedicated web interfaces

Registry-registrar relationship: the web interfaces

Web interfaces:

- Wide accessibility highly desirable
- EPP based systems

Usually, divided in two sections:

- Public information
- Restricted information for registrars or ISP

Relational databases

Why use a database?

Look at this from the viewpoint of database use vs. a spreadsheet or flat file:

DB	Spreadsheet/FlatFile
Multi-user access.	Single user access
Speed and available complexity of queries.	Slow updates
Easy to extend.	Need script language
Keep access to your data secure.	File based protection (all or nothing)
Maintaining data integrity.	Protect cells, but fragile
Relational queries.	Formula like, not optimized for queries

What's the problem?

What types of problems are we trying to solve or avoid?

- Large zone file maintenance.
- Customer accounting.
- Customer service and tracking.
- Making sure that your data is correct.
- Keep your data secure:
  - Customer records.
  - Accounting records.

Multi-user access

1. A flat file (spreadsheet) can only be accessed by one person at a time.

2. As your organization grows you may have multiple people needing access to update records (aliases, mx records, A records, etc.).

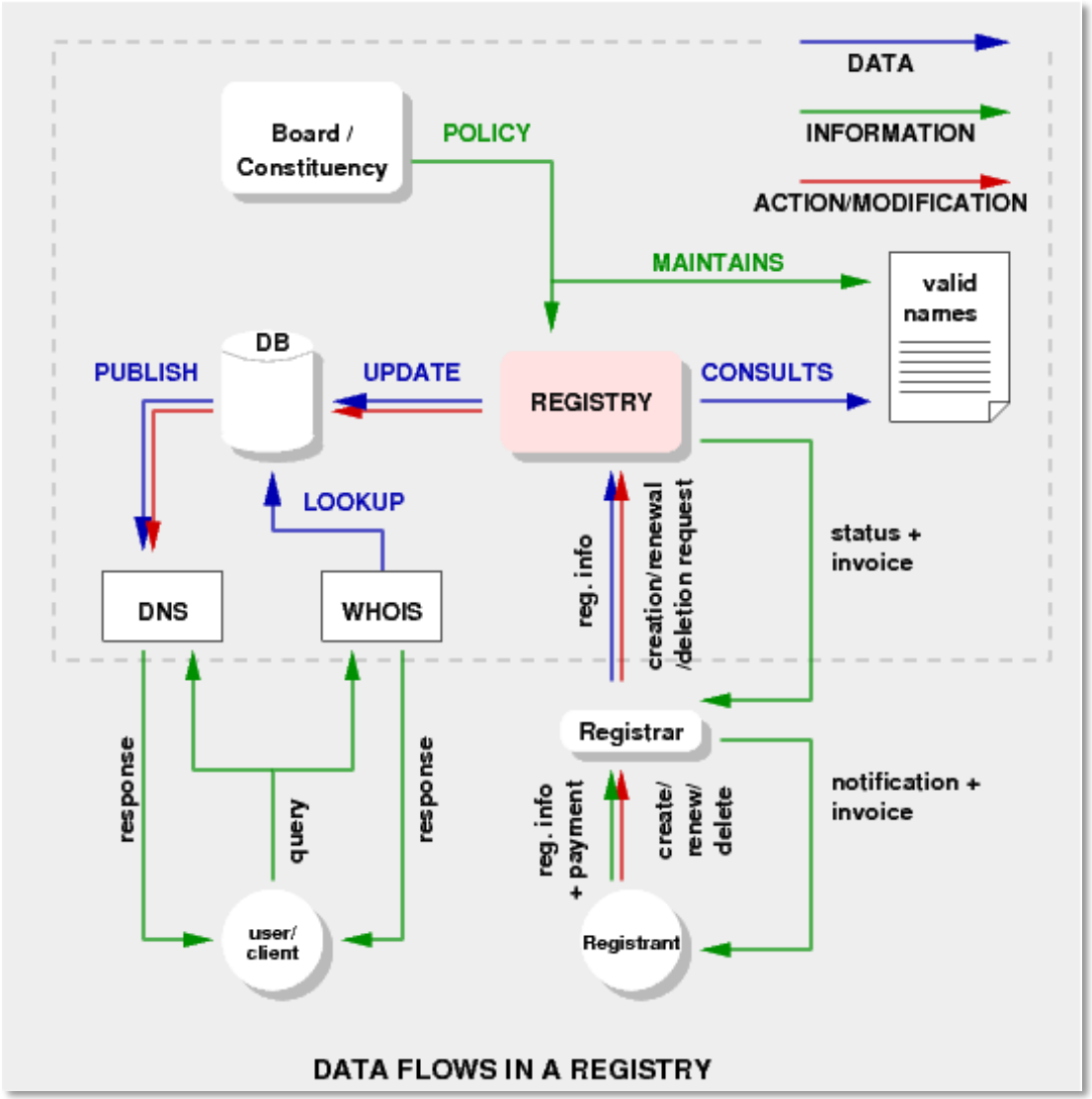
3. Multi-user access means better customer service and better efficiency, and a lowered risk of inconsistencies (simultaneous update of a registration by two employees)

Multi-user access (2)

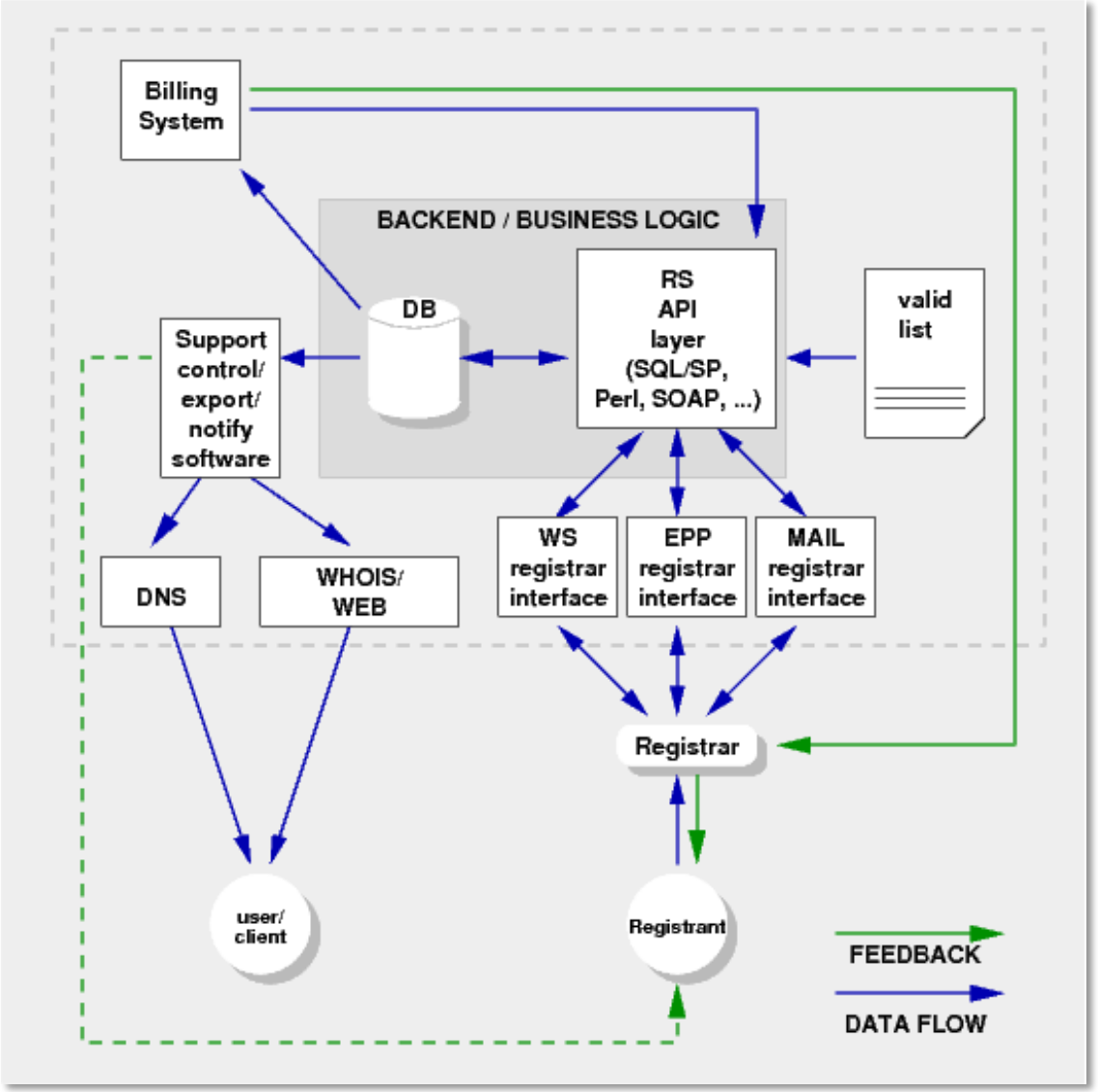
Multi-user access is a prerequisite for being able to expand the operations of the registry:

- The marketing and business development departments will want to generate reports (popular names, average registration time, etc...)
- The billing department will need to update the DB (directly or not) to mark delegations for which no renewal fee has been paid (mostly in Registry-Registrant, or 2R, models)

Reminder 1


DATA
INFORMATION
ACTION/MODIFICATION
Board / Constituency
POLICY
MAINTAINS
valid names
DB
PUBLISH
UPDATE
REGISTRY
CONSULTS
LOOKUP
DNS
WHOIS
reg. info
creation/renewal /deletion request
status + invoice
Registrar
response
query
response
reg. info + payment
create/ renew/ delete
notification + invoice
user/ client
Registrant
DATA FLOWS IN A REGISTRY

Reminder 2


Billing System
BACKEND / BUSINESS LOGIC
DB
RS API layer (SQL/SP, Perl, SOAP, ...)
valid list
Support control/ export/ notify software
DNS
WHOIS/ WEB
WS registrar interface
EPP registrar interface
MAIL registrar interface
Registrar
user/ client
Registrant
FEEDBACK
DATA FLOW

DB: Easy to extend

Multiple users accessing zone file information via a database:

- Now you can create a programatic interface to generate your zone file.
- Zone file can be generated at regular intervals without human intervention.
- Database can ensure that data entered is unique to create correct zone files.
- The DB Schema itself can be modified to accommodate changes

DB: Maintaining data integrity

- You want to know that your data is not corrupt and you want to keep it that way.
- A well-designed database can help “force” your organization to enter correct data.
- A database can verify data relations and integrity of your data.
- Databases have many tools for backup, recovery, cleanup, and data checking.

DB: Relational Queries

- This is something that you cannot do in a spreadsheet. Queries are limited.
- A relational database lets you create multiple tables with records, and connect these.
- You can view your data in many different ways.
- Finding relations, querying for them, and getting results is an *extremely* powerful feature of relational databases.

Speed and complexity of queries

A well-designed database allows for extremely fine-grained queries on very large sets of data. These queries are:

- Fast!
- You can mathematically guarantee the correctness of queries using boolean logic.
- You can guarantee completeness of results.
- And, did I say the queries were “fast!”...

Public databases

By *public databases* we mean:

- Database software that is available under “free” licenses.
- Database systems developed in a public forum.
- Commercial databases must be purchased.
- Commercial databases require you to pay for newer versions.
- Both public and commercial databases have support contracts that you can pay for.
- Public databases have a legacy of user community support that is very effective.

Some Database Choices

Public databases

- MySQL: www.mysql.org*

- PostgreSQL: www.postgresql.com

- MiniSQL: www.hughes.com.au

Some “not” public databases

- Oracle: www.oracle.com

- IBM’s DB2: www.ibm.com/db2

- Microsoft SQL: www.microsoft.com/sql

MySQL and PostgreSQL

Religious wars have been started over the question, “Which is better?”

versus

One general opinion (imho) goes like this:

PostgreSQL has more advanced database features and is more complete while MySQL has a huge developed base of applications, is easier to use, and is very fast for small to medium sized db's.

MySQL and PostgreSQL cont.

- Both are available for Linux and FreeBSD.
- Both are free.
- Both have tools for administering them graphically.
  - pgAdmin and phpPgAdmin, etc.
  - MySQL Workbench, phpMyAdmin, etc.
  - Lots more for both, including web-based tools.
- Both can be accessed from your favorite programming language.
- Both are used to create dynamic web sites.

Some Flies in the Ointment…

MySQL appears to be in trouble (May 2010):

- Oracle bought MySQL (via Sun) in 2009
- MySQL core developers and founders have left the project (before the Oracle purchase)
- Oracle bought MySQL's query engine (innodb)
- MySQL competes with Microsoft SQL Server (low to mid-range market), so maybe there's incentive to keep it around, but who will develop future MySQL releases?
- MySQL has been forked several times.
- MySQL "6.0" release no where to be seen…

Acronyms!

"LAMP"

– Linux, Apache, MySQL, Php

"FAMP"

– FreeBSD, Apache, MySQL, Php

"LAPP"

– Linux, Apache, PostgreSQL, Php

"FAPP"

– FreeBSD, Apache, PostgreSQL, Php

etc…

Types of data to store

Customer:

- Accounting records
- Transactions
- Support

Zone file:

- Domain records

Relations:

- Customer
- Domains

Generating a Zone file from a database

- **Your choice of language:**
  - PHP, Perl, Python, ...
  - C, C++
  - etc...

- Need to generate a valid zone
- Validation of data entering the DB **and** leaving it
- Look through all records (ensures completeness).
- Built dynamically so you can still be accessing your zone and customer data at the same time.

Sample schema

```
create table data (
  zone      text,     - ”nsrc.org.”
  name      text,     - ”www”
  ttl       text,     - ”3600”
  rdtype    text,     -  ”A”
  rdata     text,     -  ”128.223.157.19”
  locked    bool,     - ”t”
  comments  text,     - ”Website for NSRC.org”
  dynamic   bool,     - ”f”
);
```

Sample schema

Note that the zone file contains only a *subset* of the data to be found in the DB.

For instance, registrant information, comments, date of registration, etc... are not exported to the zone file.

Likewise, a WHOIS server will not show zone data – possibly which NSes are published for a given zone.

Some registry tools (free to use)

CoCCA: http://sourceforge.net/projects/coccaopenreg/
Consortium, Council of Country Code Administrators

CodevNIC: http://codev-nic.generic-nic.net/
.fr project, Co-developped NIC

DNRS: http://sourceforge.net/projects/dnrs/
.nz, Domain Name Registry System

FRED: http://fred.nic.cz/
.cz, Free Registry for ENUM and Domains

Some registry tools (free to use)

CoCCA: http://sourceforge.net/projects/coccaopenreg/
Consortium, Council of Country Code Administrators

CodevNIC: http://codev-nic.generic-nic.net/
.fr project, Co-developped NIC

DNRS: http://sourceforge.net/projects/dnrs/
.nz, Domain Name Registry System

FRED: http://fred.nic.cz/
.cz, Free Registry for ENUM and Domains

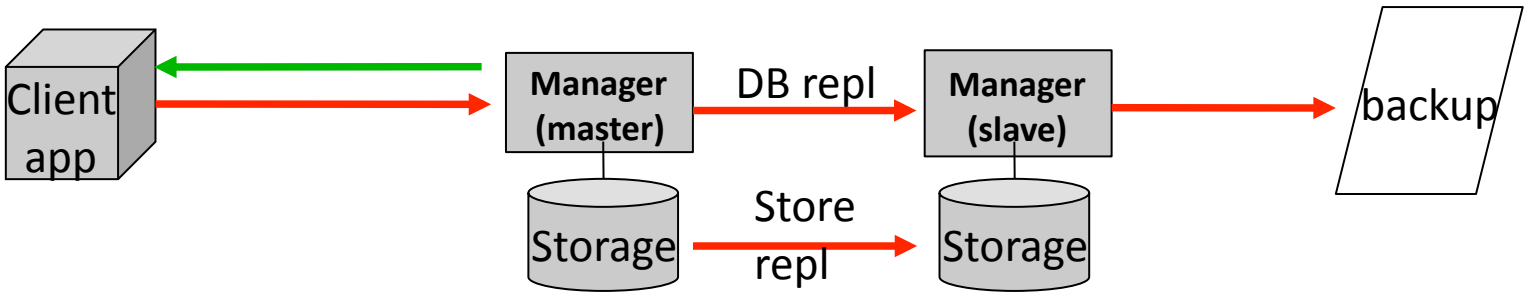
Reliable, Robust and Resilient Registry Operations

Database availability

Databases are more complex than flat text files or spreadsheets

Corruption, operational errors, vandalism: all can render the DB unusable

Backup is of course a must, but consider DB replication or storage replication to minimize downtime:



Registry Services availability

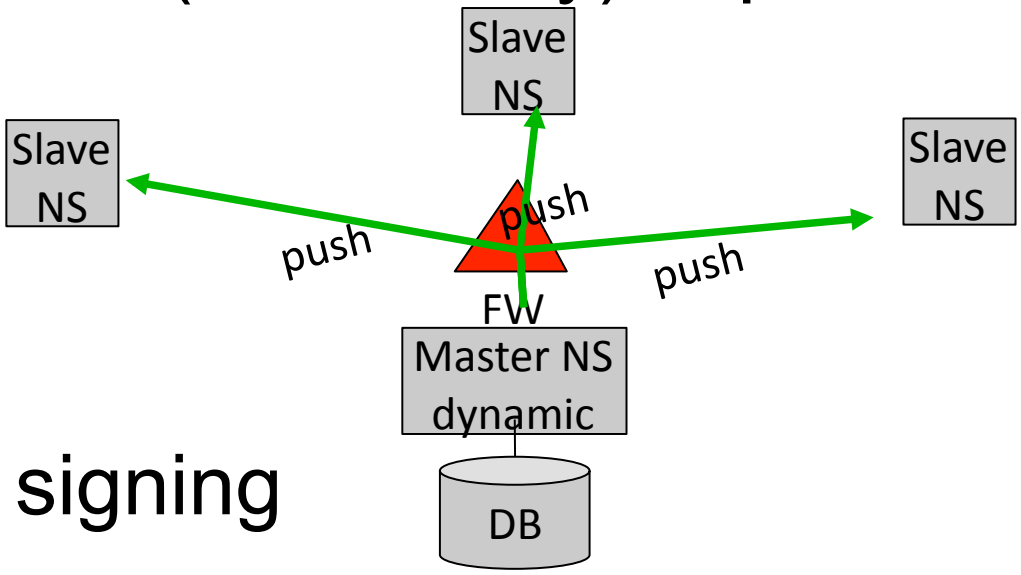
- It is tempting to use a DB and perform ”real time” publication of the zone and WHOIS
- There are inherent risks to this, such as outline in the previous slide
- DB is weakest link (complex systems fail more often than simple ones), versus the nameserver or the WHOIS service
- Consider decoupling zone storage and data publication

Registry Services availability (2)

- For instance, re-publish the zone file at regular intervals
- This fits well with DNSSEC, where zone RRs need to be regularly resigned
- The same goes for WHOIS data
- Having a separate process generating the zone file and WHOIS from the DB, means that these services can continue to run, in the worst case scenario that the DB crashes

Registry Services availability (3)

Another solution is to have a ”hidden master” setup. It can itself be directly linked to the DB, but public-facing nameserver instances are slave (Secondary) copies of the master.



Zone signing (DNSSEC) is another factor to consider.

Registry Services availability (4)

- Registration services need not fail entirely either.
- Protocols like EPP do imply having access to a DB for modification of registrations
- But it is still possible to build the service handling creation to queue the requests until the DB becomes available
- It is a matter of policy to decide which approach to adopt

Maintaining a trail of changes

- When the zone is a text file or a spreadsheet, it's fairly easy to keep track of changes to the zone:
  - Make a copy of the zone with a timestamp everytime you make a change
  - Can be automated.
- How does one do this with a DB?
  - When exporting the zone to a text file, version it (we'll see this) or implement versioning in the DB (more complex, but extremely useful)
  - What granularity?

Anycasting DNS servers

DNS services

Many Name servers for a ccTLD

- For redundancy
- Traffic load
- Better response time

Secondary servers must be placed at both topologically and geographically dispersed locations on the Internet

- RFC 2182
- AfriNIC, RIPE NCC, ISC, PCH, PSG.COM, NSRC,etc... provide secondary DNS services for ccTLDs

Anycasting of DNS service

Benefits of anycast

- Transparent fail-over redundancy
- Load balancing
- Latency reduction (“nearest” instance is ”picked”)
- Attack mitigation (difficult to take down many instances)
- Configuration simplicity (for end users) (for example using a single IP for recursive service)

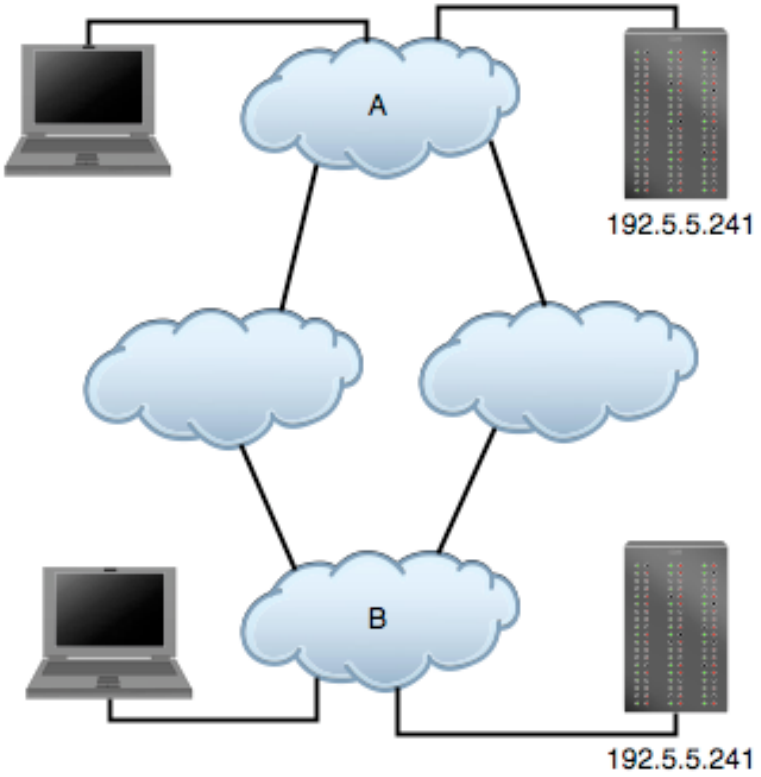
Some of listed secondary service providers in the previous slide run Anycast.

There are some commercial anycast DNS service providers.

Anycast DNS servers

- Has a single IPv4 or IPv6 address
- Requests sent to these addresses are routed to different nameservers, depending on the origin of the request
- This behavior is transparent to devices which send requests

Anycast Routing

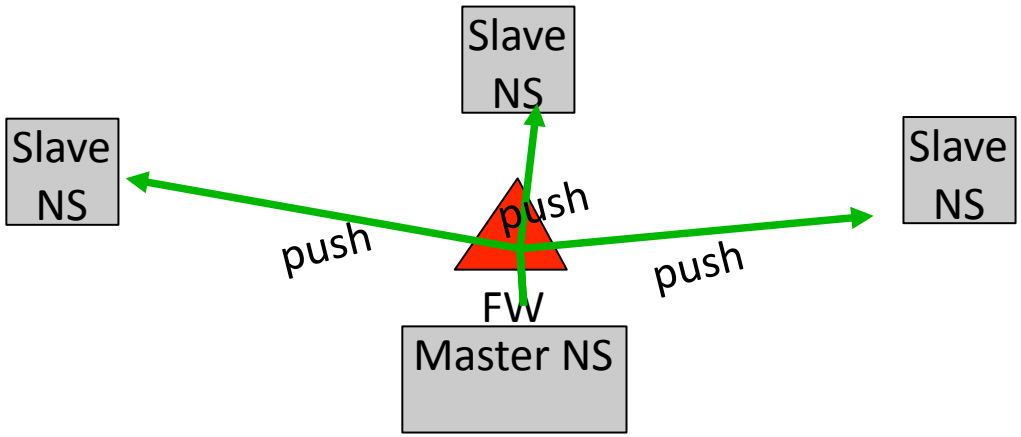


Hierarchical Anycast

- Some Anycast nodes provide service to the entire Internet (global nodes)
  - very large, well-connected, secure and over-engineered nodes
- Others provide service to a particular region (local nodes)
  - Smaller
- Each local node's routing is organised such that it should not, under normal circumstances, provide service for clients elsewhere in the world – attract local traffic

Integrity of secondary zone data

Reminder



Consider securing communications between master and slaves via TSIG – or use a secure replication mechanism like SSH/rsync to push the zones.

Questions?