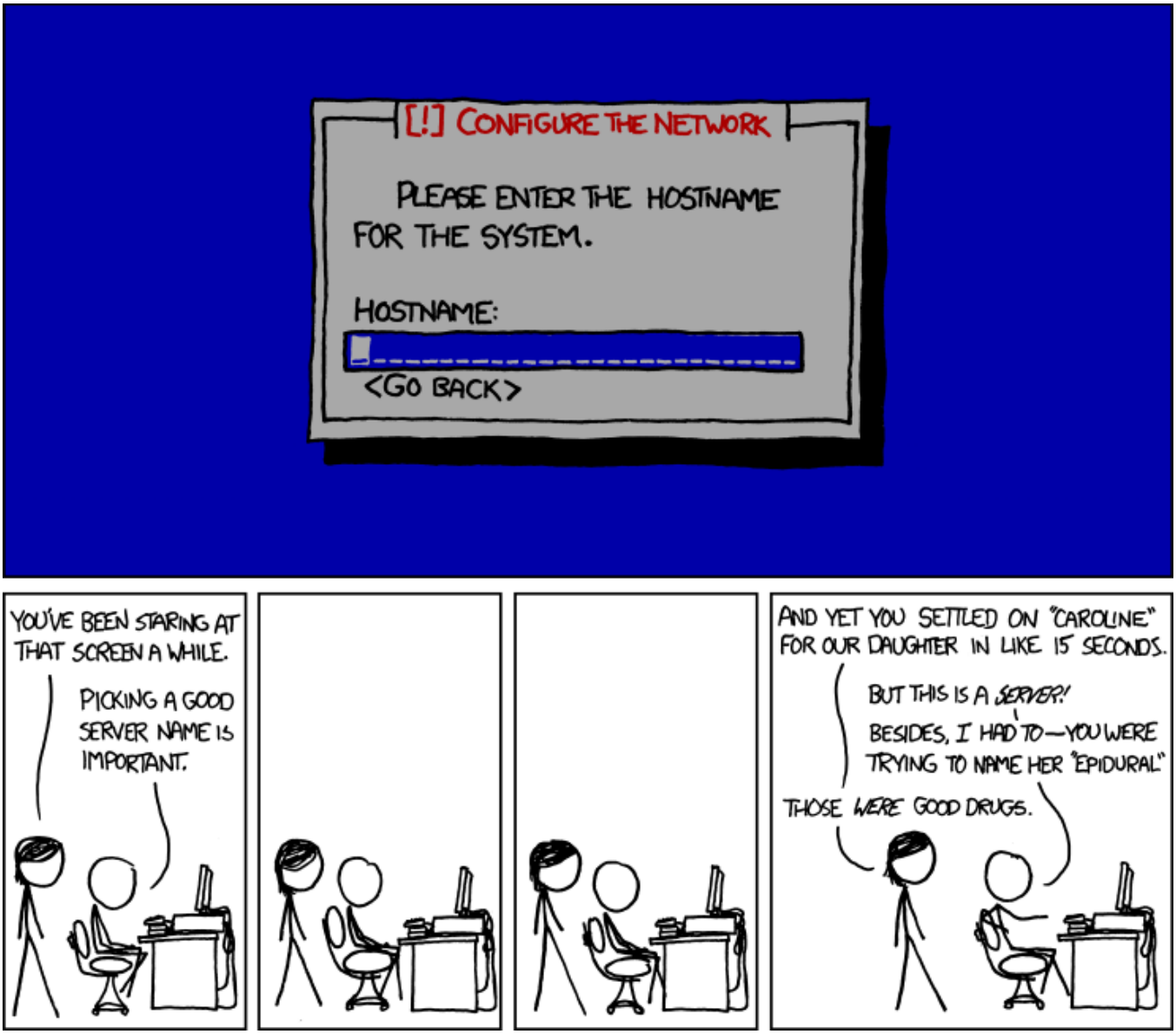
A Proper Server Naming Scheme

Here at MNX, we've been busy setting up a brand new data center for our cloud hosted services. We started off as a consulting company providing managed Linux services, which means we have been exposed to a ton of different customer environments and an equal number of schemes for naming equipment...not all of them good. It's a problem that goes back as far as computers have existed, and everyone has their own opinion on the "best" way to name hosts. Most methods start out fine at the beginning, but quickly become unwieldy as infrastructure expands and adapts over time.

Since we're starting fresh with this data center, we wanted to come up with our own naming scheme to address the common problems we've seen elsewhere. We gleaned ideas from numerous sources such as data published by large-scale companies, various RFC's on the topic, and blog/forum posts a'plenty. Taking all of that into account, we've developed some best practices that should work for most small-to-medium businesses naming their own hardware.

XKCD seems to have something for every topic



First, I'll go over the naming scheme then cover some of the finer points and justification for our choices

A Records

To start off, name each host (via the appropriate method for your operating system) and set its DNS A record to a randomly chosen word pulled from a list:

```
crimson.example.com.    A    192.0.2.11
```

There are many pools of words to choose from, but the specific word list we

recommend comes from Oren Tirosh's mnemonic encoding project. These 1633 words were chosen very specifically to be short (4-7 letters), phonetically different from one another, easy to understand over the phone, and also recognizable internationally. The mnemonic word list should be much less prone to typos and transposed characters when compared to more structural names. A lot of time and research went into these words, and their properties make them ideal for our purpose.

Essentially, the hostname should not have any indication of the host's purpose or function, but instead acts as a permanent, unique identifier to reference a particular piece of hardware throughout its lifecycle (try not to reuse names when hardware dies). This name should be used to physically label the equipment and will mostly be useful to operations engineers, remote hands, and for record keeping. It's also what the reverse DNS PTR record should resolve to.

CNAME Records

Next, assign one or more DNS CNAME records to cover useful functional details about the machine such as geography, environment, work department, purpose, and so on. This is all information that will be mirrored in your CMDB and easily referenced.

The CNAME records are what developers should know and use for interconnecting services. Keeping the structure of these names consistent will lower the mental effort necessary to remember a hostname when you need it...

Standardized CNAME Structure

Start with your registered domain, and segment each piece of additional information as a proper subdomain going down from there. DNS is hierarchical by design, so taking advantage of that will provide us with some benefits later on.

```
<wip>.example.com.   CNAME   crimson.example.com.
```

Specify Geography

After your domain name, add a subdomain referencing the geography of the host. Use the 5-character United Nations Code for Trade and Transport Locations (UN/LOCODE) value based on the address of the host's data center. It covers more specific locations than something like the IATA airport codes, and is still a well defined standard.

In most cases, you can drop the 2-character country code portion and just use the remaining 3-character location code. That is, unless you have data centers in multiple countries and the locations happen to use conflicting codes, just use *nyc.example.com* and not *nyc.us.example.com*.

```
<wip>.nyc.example.com.   CNAME   crimson.example.com.
```

Specify Environment

Next up, specify the environment that the host is a part of:

- `dev` – Development
- `tst` – Testing
- `stg` – Staging
- `prd` – Production

These should be based on whatever process model you follow for release management...you may have more or less designations as well as environments like sandbox, training, etc..

```
<wip>.prd.nyc.example.com.   CNAME   crimson.example.com.
```

Specify Purpose and Serial Number

Last up, specify the basic category of the host's function and append a serial number:

- `app` – Application Server (non-web)
- `sql` – Database Server
- `ftp` – SFTP server
- `mta` – Mail Server
- `dns` – Name Server
- `cfg` – Configuration Management (puppet/ansible/etc.)
- `mon` – Monitoring Server (nagios, sensu, etc.)
- `prx` – Proxy/Load Balancer (software)
- `ssh` – SSH Jump/Bastion Host
- `sto` – Storage Server
- `vcs` – Version Control Software Server (Git/SVN/CVS/etc.)
- `vmm` – Virtual Machine Manager
- `web` – Web Server

For the serial number, use zero-padded numbers based on your expected capacity. Plan for expansion, but usually two digits will be more than sufficient.

```
web01.prd.nyc.example.com.   CNAME   crimson.example.com.
```

Increment the serial numbers sequentially and segment them based on the type of server in a particular data center, rather than a globally-unique index. That means you may have a *`web01`* in multiple data centers.

Convenience Names

Beyond the standardized structure, you may want additional CNAME records for convenience words like *webmail*, *cmdb*, *puppet*, etc..

```
webmail.example.com.   CNAME   crimson.example.com.
```

Special Cases

Networking and Power Equipment

For networking and power equipment, the hardware dictates the purpose and it's not likely that you can just move them without reconfiguration. Knowing that, ignore the random word naming convention and use functional abbreviations for the DNS A record itself:

- con – Console/Terminal Server
- fwl – Firewall
- lbl – Load Balancer (physical)
- rtr – L3 Router
- swt – L2 Switch
- vpn – VPN Gateway

- pdu – Power Distribution Unit
- ups – Uninterruptible Power Supply

…you'll probably want the data center geographical info in there as well. You can still add CNAME records for more specific info like core/dist, public vs private, etc. if desired.

```
rtr01.nyc.example.com.   A   192.0.2.1
```

Secondary and Virtual IP Addresses

The tricky part with secondary and virtual IPs (used for high availability, web services, network migrations, VLAN tagged traffic, etc.) are that they might be floating and not tied to a specific piece of hardware. That being the case, it's easiest to just assign the functional name directly to the DNS A record and follow the normal naming convention.

Mail and Name Servers

For your mail and name servers, you have to utilize DNS A records since MX and NS records must never point to a CNAME alias. That said, you can have more than one DNS A record, so stick with the regular scheme and add something else for the public MX and NS records to utilize.

```
puma.example.com.     A   192.0.2.20
mta01.example.com.    A   192.0.2.20
```

DNS Configuration

Since we used proper DNS subdomains for each unit of data, we can set the search domains on each host to only pay attention to their own local category of machines:

```
search  prd.nyc.example.com  example.com
```

This makes it convenient when working on the machines, as you can use the shorter version of hostnames to, for instance, *`ping sql01`* rather than having to type in the full *`ping sql01.prd.nyc.example.com`* when communicating within a data center.

In general, our naming scheme also allows you to prevent inadvertent information disclosure by publicly exposing only the short random hostname while resolving the functional names solely on the internal network. It's a bit of security through obscurity, but something to consider. (Note, you'd have to tweak the "special cases" naming convention if you want to hide those as well)

Private Network and Out-of-Band Addressing

You can also take advantage of internal DNS resolution to expose private

network addresses and out-of-band/IPMI/iDRAC addresses. The domains should match the other records, but once again, utilize a proper subdomain. Note that best practices dictate not using a fake TLD, as ICANN could register them at any time and combining networks becomes trickier.

Complete Naming Scheme Example

```
crimson.example.com.          A       192.0.2.11
crimson.lan.example.com.      A       10.0.2.11
crimson.oob.example.com.      A       10.42.2.11
web01.prd.nyc.example.com.    CNAME   crimson.example.com.

melody.example.com.           A       192.0.2.12
melody.lan.example.com.       A       10.0.2.12
melody.oob.example.com.       A       10.42.2.12
web02.prd.nyc.example.com.    CNAME   melody.example.com.

verona.example.com.           A       192.0.2.13
verona.lan.example.com.       A       10.0.2.13
verona.oob.example.com.       A       10.42.2.13
cfg01.prd.nyc.example.com.    CNAME   verona.example.com.
mon01.prd.nyc.example.com.    CNAME   verona.example.com.
puppet.example.com.           CNAME   verona.example.com.
nagios.example.com.           CNAME   verona.example.com.

banjo.example.com.            A       192.0.2.104
banjo.lan.example.com.        A       10.0.2.104
banjo.oob.example.com.        A       10.42.2.104
web01.dev.pdx.example.com.    CNAME   banjo.example.com.
martinlutherkingsr.melblanc.kugupu.stevejob.kenkesey.music.filmhistory
```

Capacity

This naming scheme will easily support 1500+ global servers. If you have more servers than that, you could add in the geography portion for the random names and then reuse words from the list. The downside being

that `crimson.nyc.example.com` might have a completely different purpose than `crimson.pdx.example.com`, so there's a bit of a mental barrier. Alternatively, you could expand the initial word list, trying to add words similar in spirit to the mnemonic encoding words.

If you're managing 10,000+ servers, the host is much more likely to only have a single segmented purpose, so ignore everything we've written above and just go with a location-based or functional naming scheme.

Tips & Tricks

- You should remove potentially confusing words like *'email'* from the mnemonic encoding word list if it's technical jargon for your environment.
- Keep the purpose abbreviations consistent in length and always have the serial number padding match (i.e. don't have `01` some places and just `1` in others, always use the longer `01` for everything).
- The actual purpose abbreviations you use aren't important, just pick a scheme, make sure it's documented, and stick to it.
- It's easiest to keep the purpose abbreviations somewhat generalized, as more detailed information can be pulled from your CMDB.
- No matter what, all info should be in a CMDB and easily accessible!
- Set multiple CNAME records when logical, but keep in mind that the more records you have, the more there will be to maintain.
- Automate as much of this as possible.
- We have written a short script named genhost that can help you to randomly pick and keep track of the words you've used for hostnames.

Conclusion

Our server naming scheme lowers the mental effort required to keep track of machines and makes connecting services and maintaining proper hardware records straightforward. The aspects of a machine that are likely to change over time are contained only within the CNAME records. That

means if a server dies, you don't have to go and update all references to that host on other machines, as you can just update the CNAME records to point to a new host altogether. While our scheme does add some complexity up front, it strikes a good balance between usability, maintainability, and support for long-term growth.

How do you name your servers?

Discuss on Hacker News