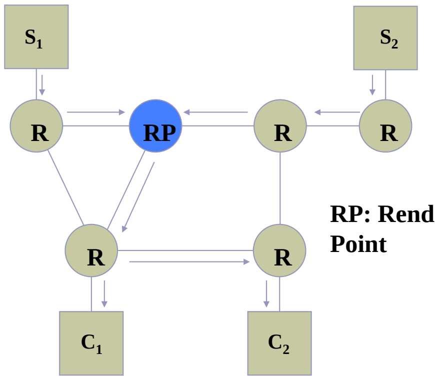
15-744: Computer Networking

L-7 Naming

Overview

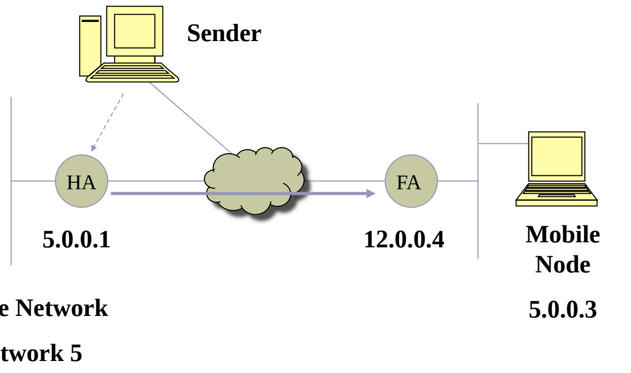
- i3
- Layered naming
  - DOA
  - SFR

Multicast



RP: Rendezvous Point

Mobility



i3: Motivation

- Today’s Internet based on point-to-point abstraction
- Applications need more:
  - Multicast
  - Mobility
  - Anycast
- Existing solutions:
  - Change IP layer
  - Overlays

So, what’s the problem?
A different solution for each service

The i3 solution

- Solution:
  - Add an indirection layer on top of IP
  - Implement using overlay networks
- Solution Components:
  - Naming using “identifiers”
  - Subscriptions using “triggers”
  - DHT as the gluing substrate

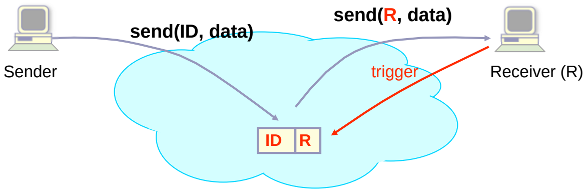
Only primitive needed

Indirection

Every problem in CS … ☺

i3: Rendezvous Communication

- Packets addressed to identifiers (“names”)
- Trigger=(Identifier, IP address): inserted by receiver



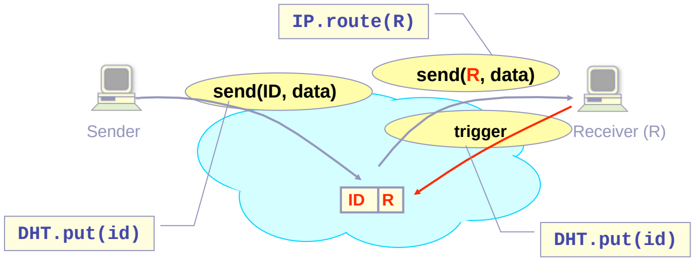
Senders *decoupled* from receivers

i3: Service Model

- API
  - `sendPacket(id, p);`
  - `insertTrigger(id, addr);`
  - `removeTrigger(id, addr); // optional`
- Best-effort service model (like IP)
- Triggers periodically refreshed by end-hosts
- Reliability, congestion control, and flow-control implemented at end-hosts

i3: Implementation

- Use a Distributed Hash Table
  - Scalable, self-organizing, robust
  - Suitable as a substrate for the Internet

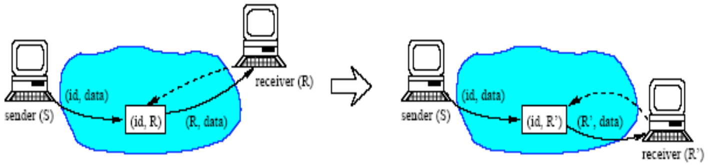


Mobility and Multicast

- Mobility supported naturally
  - End-host inserts trigger with new IP address, and everything transparent to sender
  - Robust, and supports location privacy
- Multicast
  - All receivers insert triggers under same ID
  - Sender uses that ID for sending
  - Can optimize tree construction to balance load

Mobility

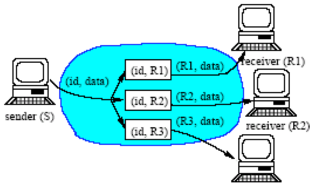
- The change of the receiver’s address
- from R to R’ is transparent to the sender



(a) Mobility

Multicast

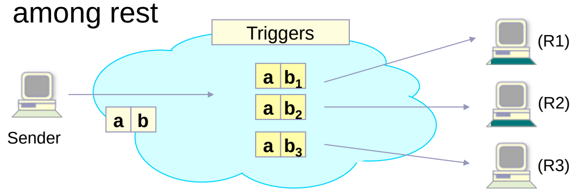
- Every packet $(id, data)$ is forwarded to each receiver R_i that inserts the trigger (id, R_i)



(b) Multicast

Anycast

- Generalized matching
  - First k-bits have to match, longest prefix match among rest



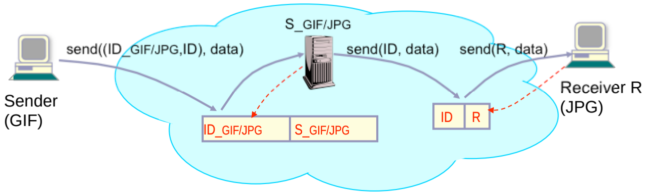
- Related triggers must be on same server
- Server selection (randomize last bits)

Generalization: Identifier Stack

- Stack of identifiers
  - i3 routes packet through these identifiers
- Receivers
  - trigger maps id to <stack of ids>
- Sender can also specify id-stack in packet
- Mechanism:
  - first id used to match trigger
  - rest added to the RHS of trigger
  - recursively continued

Service Composition

- Receiver mediated: R sets up chain and passes id_gif/jpg to sender: sender oblivious
- Sender-mediated: S can include (id_gif/jpg, ID) in his packet: receiver oblivious

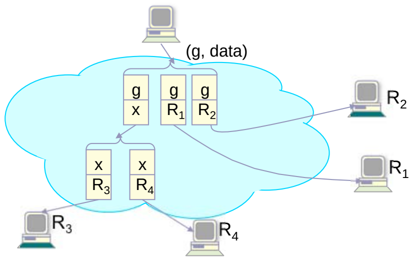


Public, Private Triggers

- Servers publish their public ids: e.g., via DNS
- Clients contact server using public ids, and negotiate private ids used thereafter
- Useful:
  - Efficiency -- private ids chosen on “close-by” i3-servers
  - Security -- private ids are shared-secrets

Scalable Multicast

- Replication possible at any i3-server in the infrastructure.
- Tree construction can be done internally



Overview

- i3
- Layered naming
  - DOA
  - SFR

Architectural Brittleness

- Hosts are tied to IP addresses
  - Mobility and multi-homing pose problems
- Services are tied to hosts
  - A service is more than just one host: replication, migration, composition
- Packets might require processing at intermediaries before reaching destination
  - “Middleboxes” (NATs, firewalls, …)

Naming Can Help

- Thesis: proper naming can cure some ills
  - Layered naming provides layers of indirection and shielding
- Many proposals advocate large-scale, overarching architectural change
  - Routers, end-hosts, services
- Proposal:
  - Changes “only” hosts and name resolution
  - Synthesis of much previous work

Internet Naming is *Host-Centric*

- Two global namespaces: DNS and IP addresses
- These namespaces are host-centric
  - IP addresses: network location of host
  - DNS names: domain of host
  - Both closely tied to an underlying structure
  - Motivated by host-centric applications

The Trouble with Host-Centric Names

- Host-centric names are *fragile*
  - If a name is based on mutable properties of its referent, it is fragile
  - Example: If Joe's Web page www.berkeley.edu/~hippie moves to www.wallstreetstiffs.com/~yuppie, Web links to his page break
- Fragile names constrain movement
  - IP addresses are not stable host names
  - DNS URLs are not stable data names

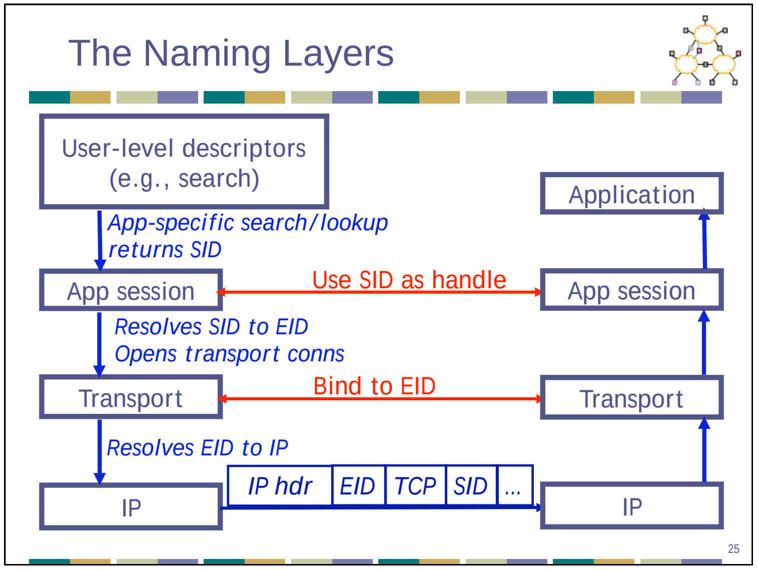
Key Architectural Questions

1. Which entities should be named?
2. What should names look like?
3. What should names resolve to?

Name Services and Hosts Separately

- *Service identifiers* (*SIDs*) are host-independent data names
- *End-point identifiers* (*EIDs*) are location-independent host names
- Protocols bind to names, and resolve them
  - Apps should use SIDs as data handles
  - Transport connections should bind to EIDs

Binding principle: *Names should bind protocols only to relevant aspects of underlying structure*



The Naming Layers

User-level descriptors (e.g., search)

App-specific search/lookup returns SID

App session

Use SID as handle

App session

Application

Resolves SID to EID
Opens transport conns

Transport

Bind to EID

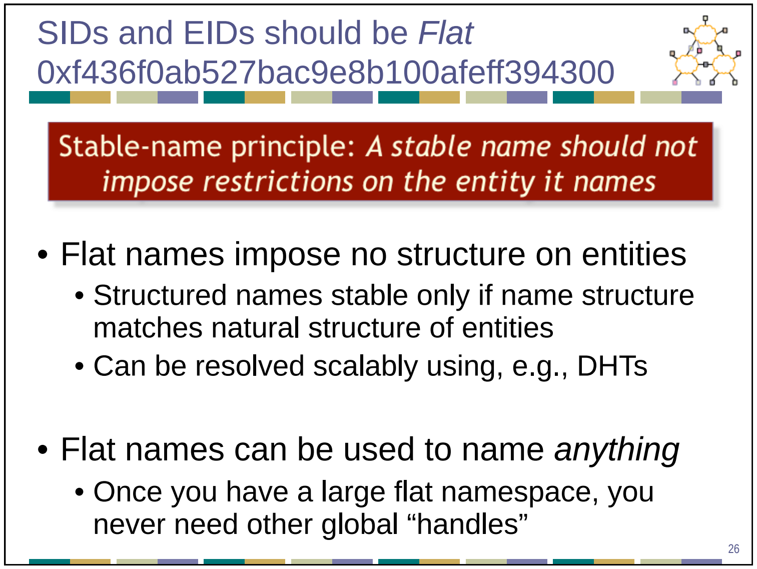
Transport

Resolves EID to IP

IP

IP hdr	*EID*	*TCP*	*SID*	...

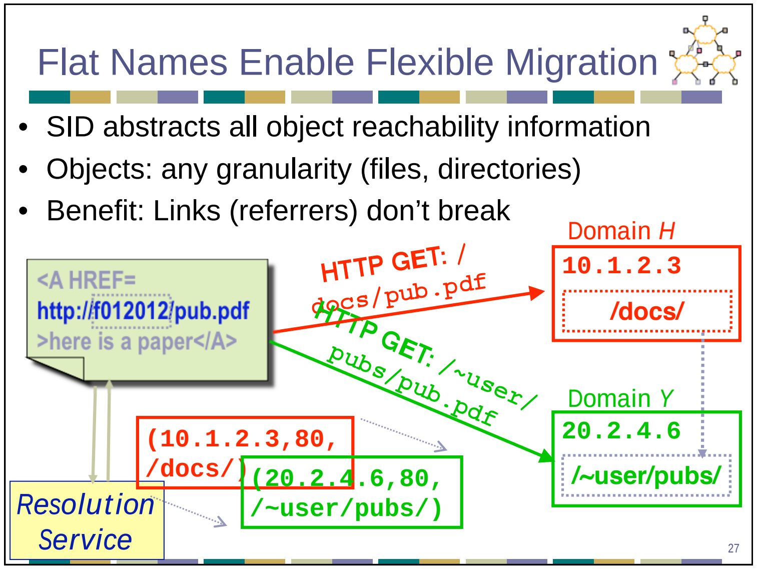
IP



SIDs and EIDs should be *Flat* 0xf436f0ab527bac9e8b100afeff394300

Stable-name principle: *A stable name should not impose restrictions on the entity it names*

- Flat names impose no structure on entities
  - Structured names stable only if name structure matches natural structure of entities
  - Can be resolved scalably using, e.g., DHTs
- Flat names can be used to name *anything*
  - Once you have a large flat namespace, you never need other global “handles”



Flat Names Enable Flexible Migration

- SID abstracts all object reachability information
- Objects: any granularity (files, directories)
- Benefit: Links (referrers) don’t break

<A HREF=
http://f012012/pub.pdf
>here is a paper</A>

HTTP GET: /docs/pub.pdf

HTTP GET: /~user/pubs/pub.pdf

Domain *H*

10.1.2.3

/docs/

Domain *Y*

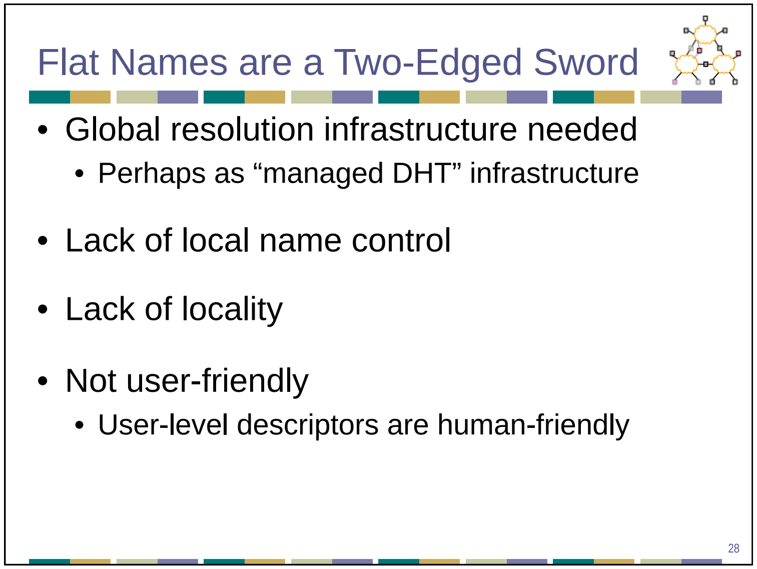
20.2.4.6

/~user/pubs/

(10.1.2.3,80, /docs/)

(20.2.4.6,80, /~user/pubs/)

Resolution Service



Flat Names are a Two-Edged Sword

- Global resolution infrastructure needed
  - Perhaps as “managed DHT” infrastructure
- Lack of local name control
- Lack of locality
- Not user-friendly
  - User-level descriptors are human-friendly

Overview

- i3
- Layered naming
  - DOA
  - SFR

Globally Unique Identifiers for Hosts

- Location-independent, flat, big namespace
- Hash of a public key
- These are called EIDs (e.g., 0xf12abc…)
- Carried in packets

IP hdr	source EID destination EID	transport hdr	body

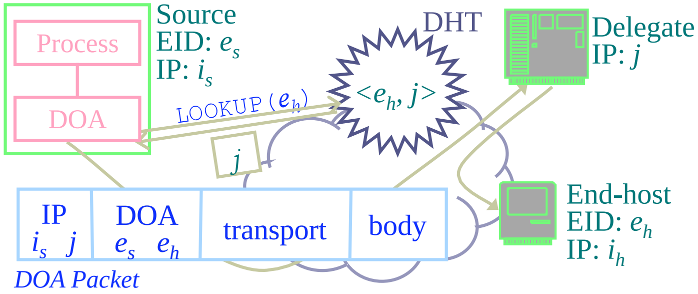
DOA hdr

Delegation Primitive

- Let hosts invoke, revoke off-path boxes
- Receiver-invoked: sender resolves receiver’s EID to
  - An IP address or
  - An EID or sequence of EIDs
- DOA header has destination stack of EIDs
- Sender-invoked: push EID onto this stack

IP hdr	source EID destination EID stack	transport hdr	body

DOA in a Nutshell

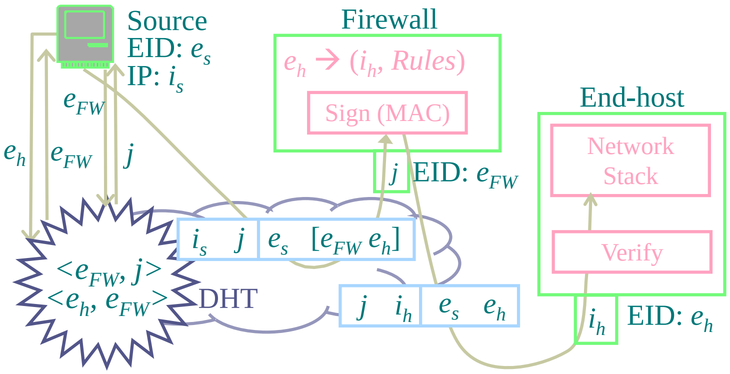


- End-host replies to source by resolving e_s
- Authenticity, performance: discussed in the paper

A Bit More About DOA

- Incrementally deployable. Requires:
  - Changes to hosts and middleboxes
  - No changes to IP routers (design requirement)
  - Global resolution infrastructure for flat IDs
- Recall core properties:
  - Topology-independent, globally unique identifiers
  - Let end-hosts invoke and revoke middleboxes
- Recall goals: reduce harmful effects, permit new functions

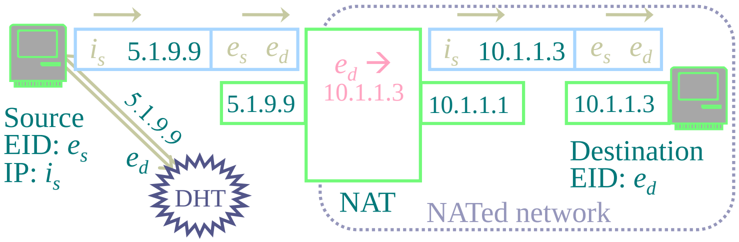
Off-path Firewall



Off-path Firewall: Benefits

- Simplification for end-users who want it
  - Instead of a set of rules, one rule:
  - "Was this packet vetted by my FW provider?"
- Firewall can be anywhere, leading to:
  - Third-party service providers
  - Possible market for such services
  - Providers keeping abreast of new applications
- DOA enables this; doesn't mandate it.

Reincarnated NAT



- End-to-end communication
- Port fields not overloaded
  - Especially useful when NATs are cascaded

Overview

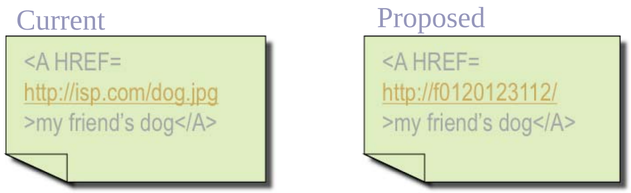
- i3
- Layered naming
  - DOA
  - SFR

Introduction

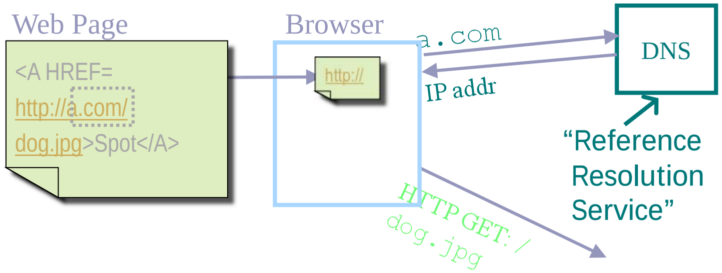
- The Web depends on linking; links contain references

  <A HREF=http://domain_name/path_name>click here</A>
- Properties of DNS-based references
  - encode administrative domain
  - human-friendly
- These properties are problems!

Web Links Should Use Flat Identifiers



Status Quo



"Reference Resolution Service"

Why not DNS?

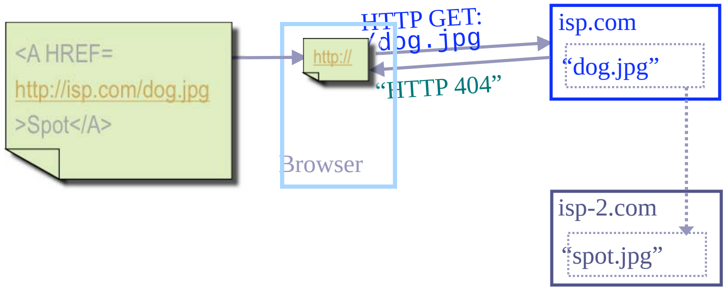
Goal #1: Stable References

Stable="reference is invariant when object moves"

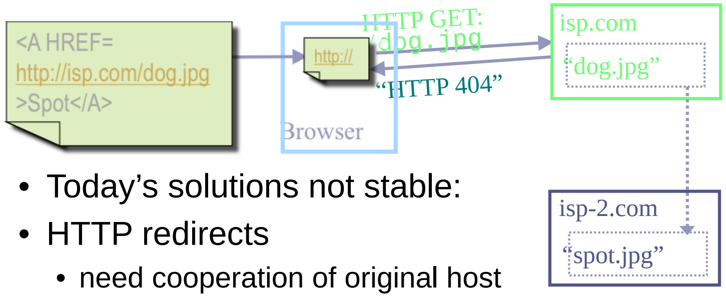
- In other words, links shouldn't break
- DNS-based URLs are not stable . . .

Object Movement Breaks Links

- URLs hard-code a domain and a path



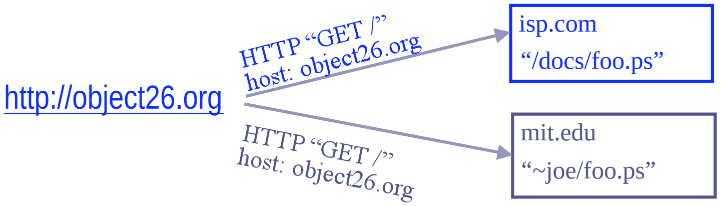
Object Movement Breaks Links, Cont'd



- Today's solutions not stable:
- HTTP redirects
  - need cooperation of original host
- Vanity domains, e.g.: internetjoe.org
  - now owner can't change

Goal #2: Supporting Object Replication

- Host replication relatively easy today
- But per-object replication requires:
  - separate DNS name for each object
  - virtual hosting so replica servers recognize names
  - configuring DNS to refer to replica servers



What Should References Encode?

- Observe: if the object is allowed to change administrative domains, then the reference can't encode an administrative domain
- What can the reference encode?
  - Nothing about the object that might change!
  - Especially not the object's whereabouts!
- What kind of namespace should we use?

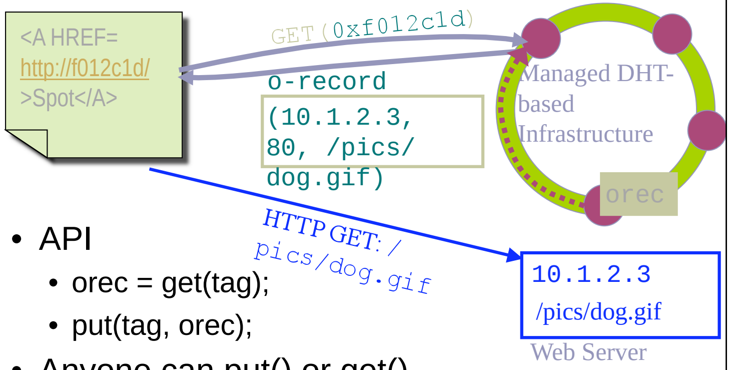
Goal #3: Automate Namespace Management

- Automated management implies no fighting over references
- DNS-based URLs do not satisfy this . . .

DNS is a Locus of Contention

- Used as a branding mechanism
  - tremendous legal combat
  - "name squatting", "typo squatting", "reverse hijacking", . . .
- ICANN and WIPO politics
  - technical coordinator inventing naming rights
  - set-asides for misspelled trademarks
- Humans will always fight over names . . .

SFR in a Nutshell



- API
  - orec = get(tag);
  - put(tag, orec);
- Anyone can put() or get()