15-744: Computer Networking

L-18 Naming

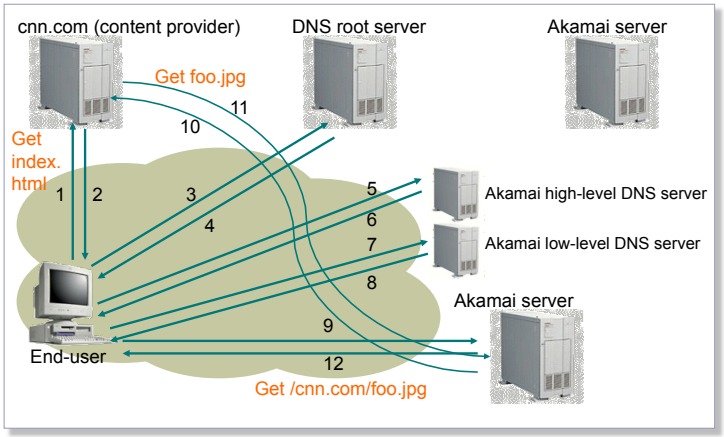
Today's Lecture

- Naming and CDNs
- Required readings
  - Middleboxes No Longer Considered Harmful
  - Internet Indirection Infrastructure
- Optional readings
  - Democratizing content publication with Coral

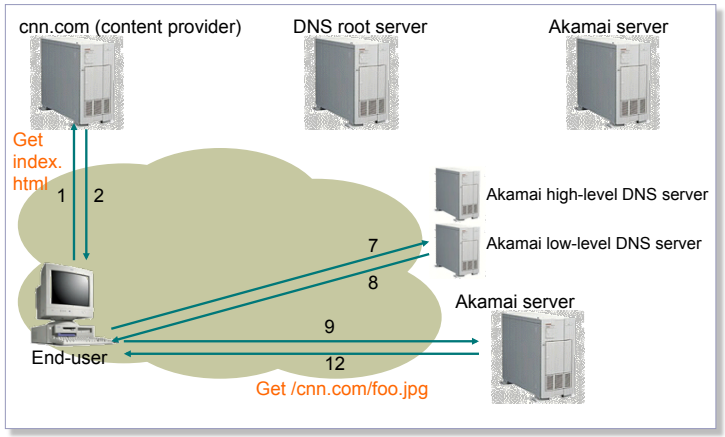
Overview

- Akamai
- i3
- Layered naming
  - DOA
  - SFR

How Akamai Works

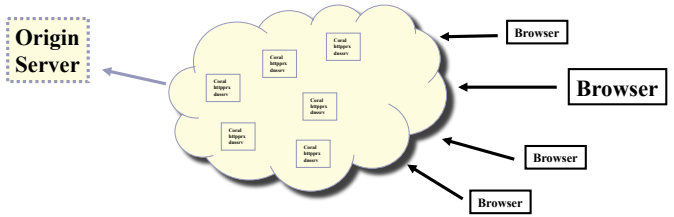


Akamai – Subsequent Requests



5

Coral: An Open CDN



Pool resources to dissipate flash crowds

- Implement an open CDN
- Allow anybody to contribute
- Works with unmodified clients
- CDN only fetches once from origin server

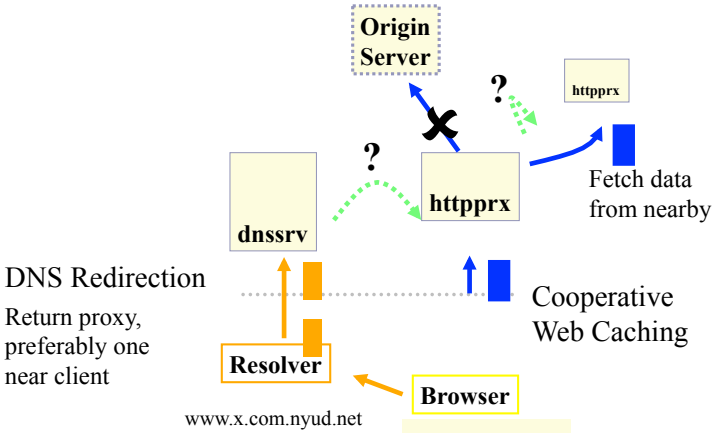
6

Using CoralCDN

- Rewrite URLs into “Coralized” URLs
- www.x.com → www.x.com.nyud.net:8090
  - Directs clients to Coral, which absorbs load
- Who might “Coralize” URLs?
  - Web server operators Coralize URLs
  - Coralized URLs posted to portals, mailing lists
  - Users explicitly Coralize URLs

7

CoralCDN components



8

Functionality needed

- DNS: Given network location of resolver, return a proxy near the client

  put (network info, self)

  get (resolver info) → {proxies}

- HTTP: Given URL, find proxy caching object, preferably one nearby

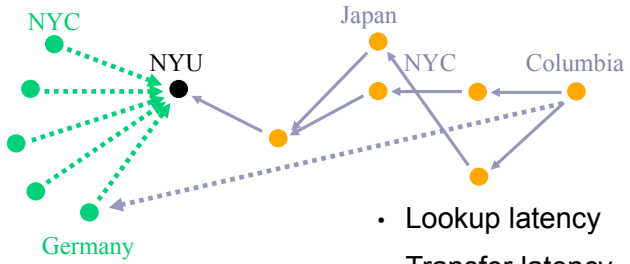
  put (URL, self)

  get (URL) → {proxies}

9

Use a DHT?

- Supports put/get interface using key-based routing
- Problems with using DHTs as given



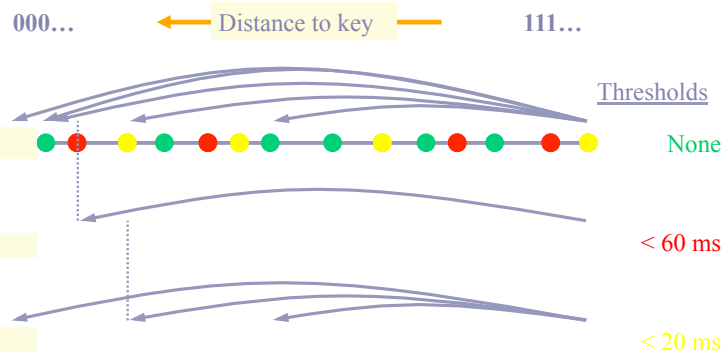
- Lookup latency
- Transfer latency
- Hotspots

10

Coral distributed index

- Insight: Don't need hash table semantics
  - Just need one well-located proxy
- put (key, value, ttl)
  - Avoid hotspots
- get (key)
  - Retrieves some subset of values put under key
  - Prefer values put by nodes near requestor
- Hierarchical clustering groups nearby nodes
  - Expose hierarchy to applications
- Rate-limiting mechanism distributes puts

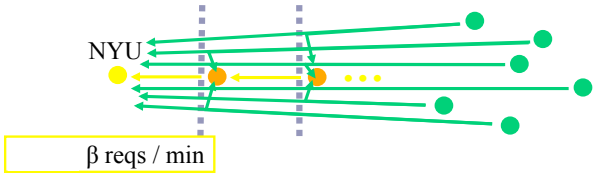
Key-based XOR routing



- Minimizes lookup latency
- Prefer values stored by nodes within faster clusters

Prevent insertion hotspots

- Store value once in each level cluster
  - Always storing at closest node causes hotspot



- Halt put routing at full and loaded node
  - Full → M vals/key with TTL > ½ insertion TTL
  - Loaded → β puts traverse node in past minute
- Store at furthest, non-full node seen

Coral Contributions

- Self-organizing clusters of nodes
  - NYU and Columbia prefer one another to Germany
- Rate-limiting mechanism
  - Everybody caching and fetching same URL does not overload any node in system
- Decentralized DNS Redirection
  - Works with unmodified clients

No centralized management or *a priori* knowledge of proxies' locations or network configurations

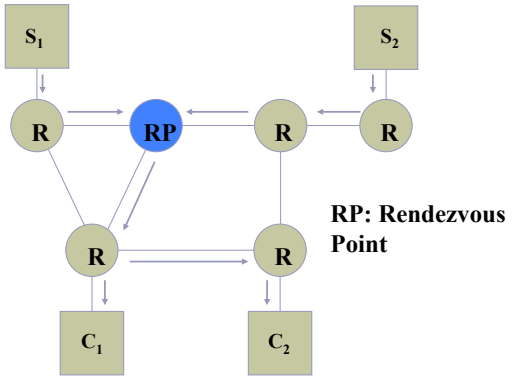
14

Overview

- i3
- Layered naming
  - DOA
  - SFR

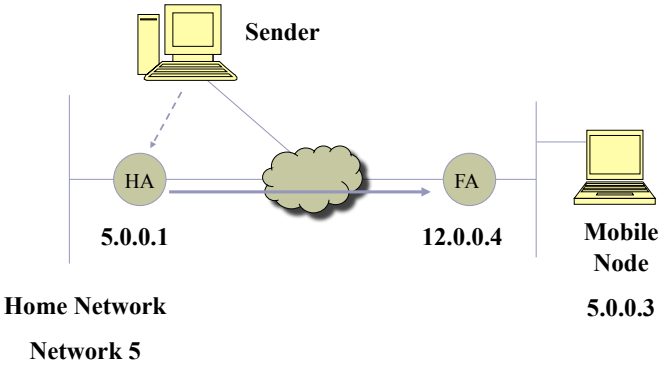
15

Multicast



16

Mobility



17

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

18

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

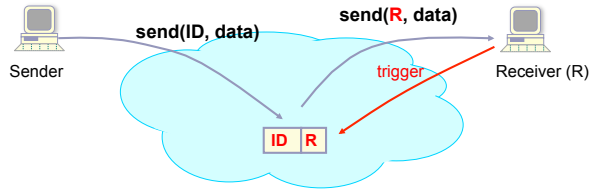
Indirection

Every problem in CS … ☺

19

i3: Rendezvous Communication

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



Senders *decoupled* from receivers

20

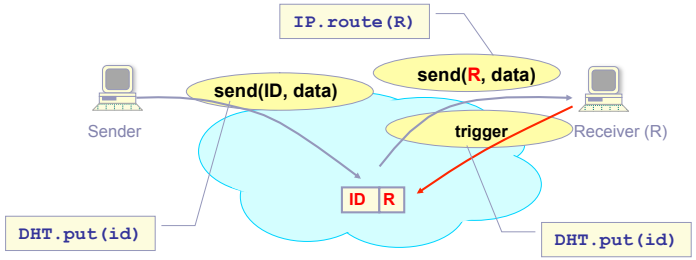
i3: Service Model

- 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

21

i3: Implementation

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



22

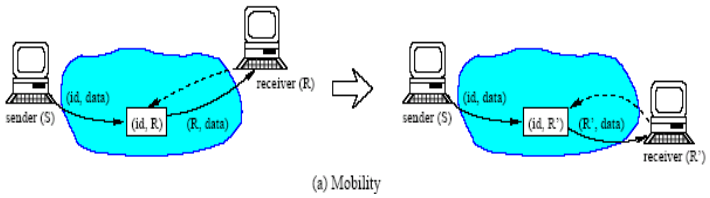
Mobility and Multicast

- Mobility supported naturally
  - End-host inserts trigger with new IP address → 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

23

Mobility

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

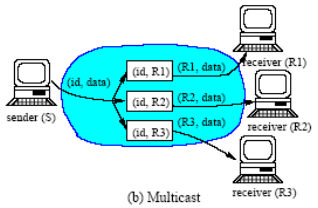


(a) Mobility

24

Multicast

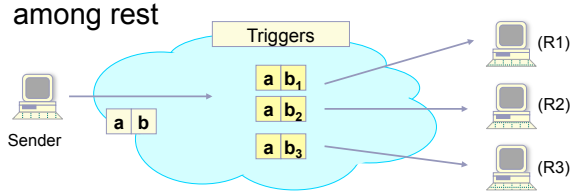
- Every packet $(id, data)$ is forwarded to each receiver R_i that inserts the trigger (id, R_i)



(b) Multicast

Anycast

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



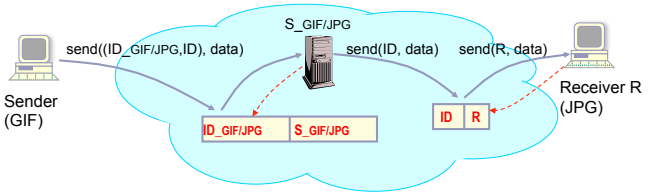
- Related triggers must be on same server
- Server selection (randomize last bits)

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



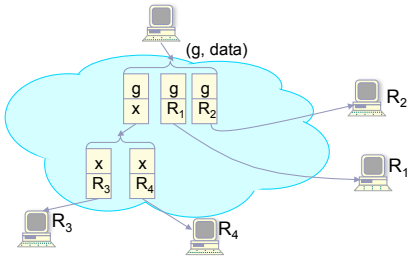
Public, Private Triggers

- 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

29

Scalable Multicast

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



30

Overview

- i3
- Layered naming
  - DOA
  - SFR

31

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, …)

32

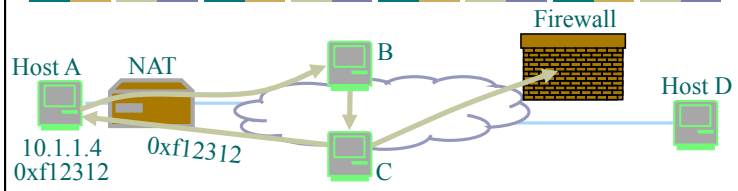
Reactions to the Problem

- Purist: can't live with middleboxes
- Pragmatist: can't live without middleboxes
- Pluralist (us): purist, pragmatist both right

- DOA goal: Architectural extension in which:
  - Middleboxes first-class Internet citizens
  - Harmful effects reduced, good effects kept
  - New functions arise

33

DOA: Delegation-Oriented Architecture



- Architectural extension to Internet. Core properties:
  1. Restore globally unique identifiers for hosts
  2. Let receivers, senders invoke (and revoke) off-path boxes: delegation primitive

34

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

35

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 application
- Such names constrain movement/replication

36

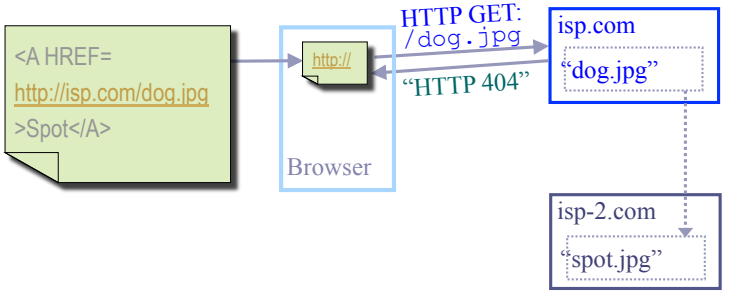
The Trouble with Host-Centric Names

- 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

37

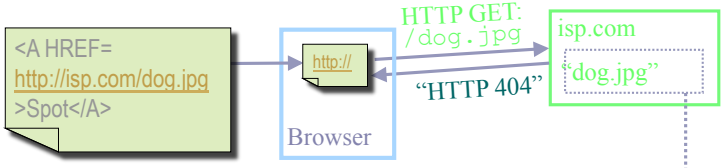
Object Movement Breaks Links

- URLs hard-code a domain and a path



38

Object Movement Breaks Links, Cont'd



- Today's solutions not stable:
- HTTP redirects
  - need cooperation of original host

39

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

40

Key Architectural Questions

- Which entities should be named?
- What should names look like?
- What should names resolve to?

41

Delegation

- Names usually resolve to "location" of entity
- Packets might require processing at *intermediaries* before reaching destination
- Such processing today violates layering
  - Only element identified by packet's IP destination should inspect higher layers

Delegation principle: *A network entity should be able to direct resolutions of its name not only to its own location, but also to chosen delegates*

42

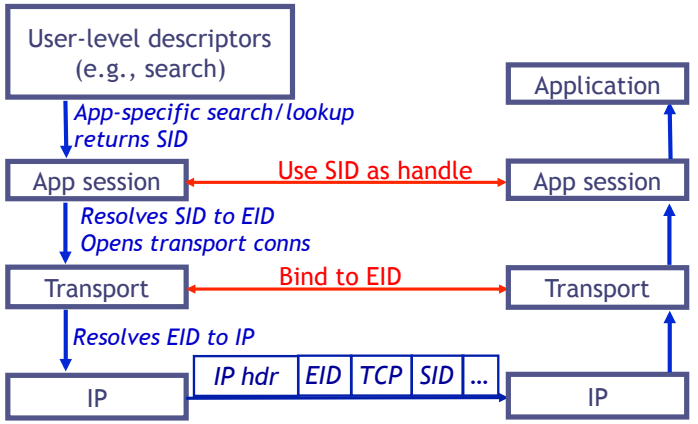
Name Services and Hosts Separately

- *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*

43

The Naming Layers



44

SIDs and EIDs should be *Flat*

0xf436f0ab527bac9e8b100afeff394300

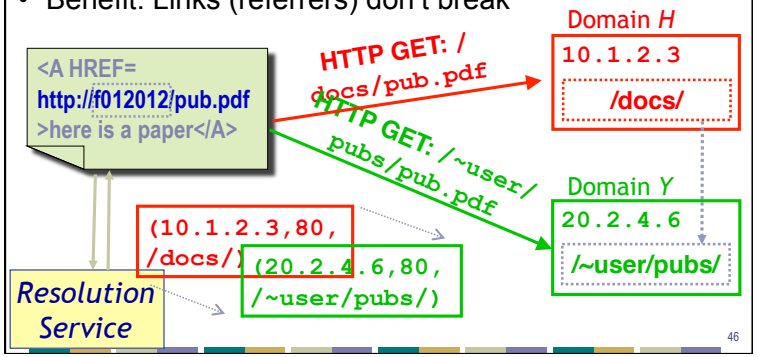
Stable-name principle: *A stable name should not impose restrictions on the entity it names*

- 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"

45

Flat Names Enable Flexible Migration

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



46

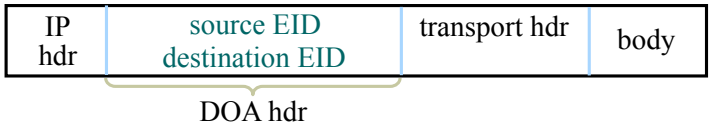
Flat Names are a Two-Edged Sword

- 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

47

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



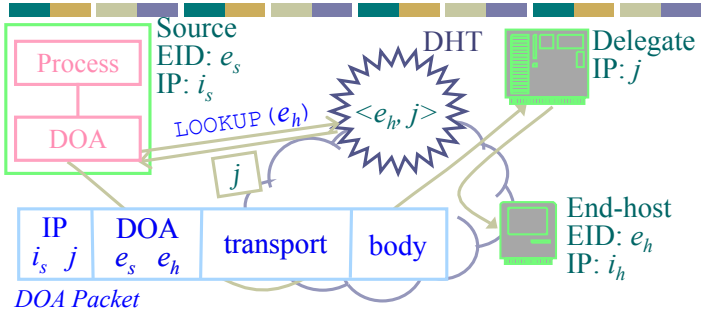
48

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

49

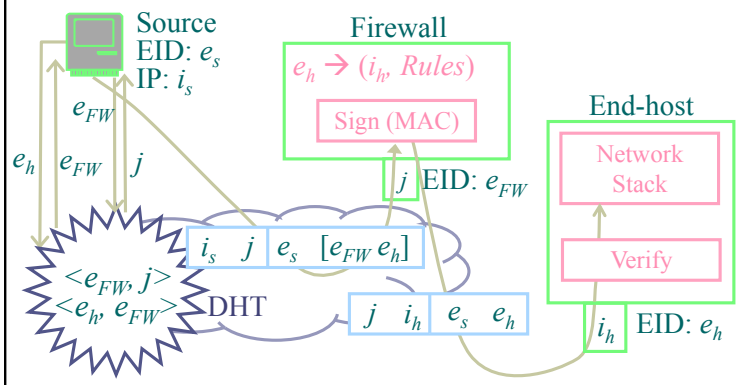
DOA in a Nutshell



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

50

Off-path Firewall



51

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.

52

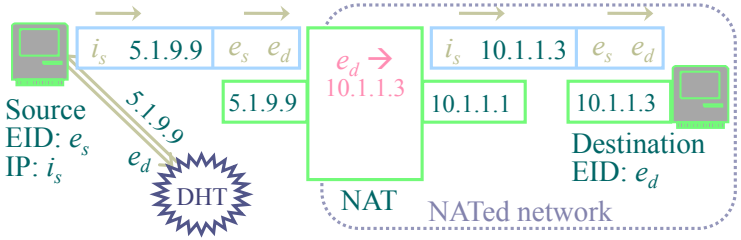
Next Lecture

- Data-oriented networking and DTNs
- Required reading:
  - Networking Named Content
  - A Delay-Tolerant Network Architecture for Challenged Internets
- Optional reading:
  - An Architecture for Internet Data Transfer
  - A Data-Oriented (and Beyond) Network Architecture

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

Reincarnated NAT



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

Key Architectural Questions

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

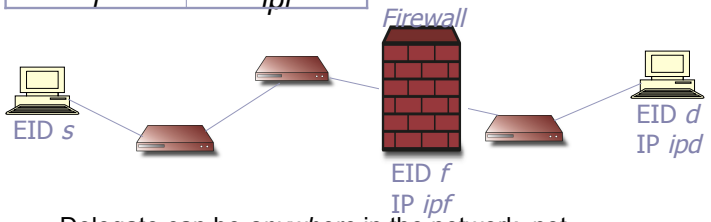
Delegation Enables Architecturally-Sound Intermediaries

Resolution svc

Dest EID	Mapping
d	*f*
f	*ipf*

Packet structure (dests only)

ipf	EID *d*	TCP hdr



- Delegate can be *anywhere* in the network, not necessarily on the IP path to *d* (*ipd*)
- SID/EID can resolve to *sequence* of delegates

57

Overview

- SFR

58

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!

59

Web Links Should Use Flat Identifiers

Current

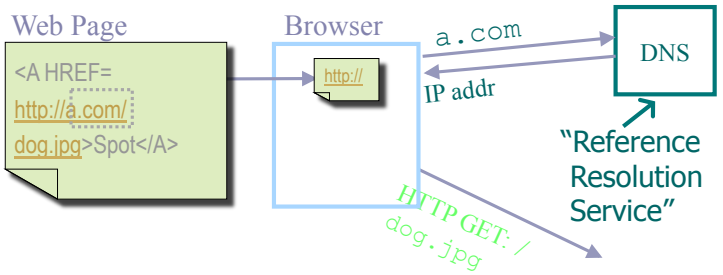
<A HREF=
http://isp.com/dog.jpg
>my friend's dog</A>

Proposed

<A HREF=
http://f0120123112/
>my friend's dog</A>

60

Status Quo



Why not DNS?

61

Goal #1: Stable References

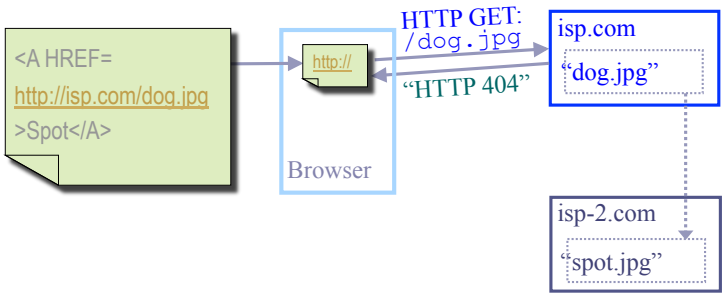
Stable="reference is invariant when object moves"

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

62

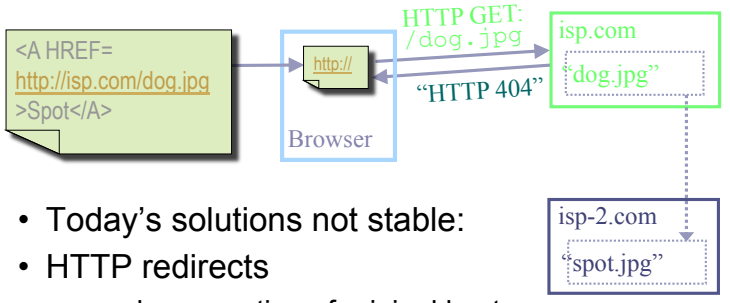
Object Movement Breaks Links

- URLs hard-code a domain and a path



63

Object Movement Breaks Links, Cont'd



- Today's solutions not stable:
- HTTP redirects
  - need cooperation of original host

64

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

65

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?

66

Goal #3: Automate Namespace Management

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

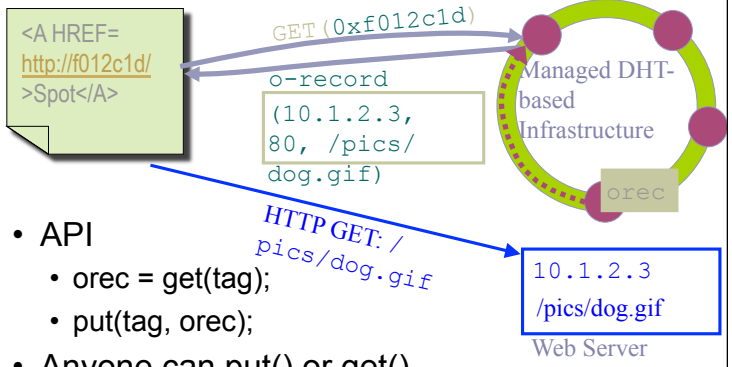
67

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 . . .

68

SFR in a Nutshell



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

Overview

- Service location

Service Location

- What if you want to lookup services with more expressive descriptions than DNS names
  - E.g. please find me printers in cs.cmu.edu instead of laserjet1.cs.cmu.edu
- What do descriptions look like?
- How is the searching done?
- How will it be used?
  - Search for particular service?
  - Browse available services?
  - Composing multiple services into new service?

Service Descriptions

- Typically done as hierarchical value-attribute pairs
  - Type = printer → memory = 32MB, lang = PCL
  - Location = CMU → building = WeH
- Hierarchy based on attributes or attributes-values?
  - E.g. Country → state or country=USA → state=PA and country=Canada → province=BC?
- Can be done in something like XML

Service Discovery (Multicast)

- Services listen on well known discovery group address
- Client multicasts query to discovery group
- Services unicast replies to client
- Tradeoffs
  - Not very scalable → effectively broadcast search
  - Requires no dedicated infrastructure or bootstrap
  - Easily adapts to availability/changes
  - Can scope request by multicast scoping and by information in request

Service Discovery (Directory Based)

- Services register with central directory agent
  - Soft state → registrations must be refreshed or the expire
- Clients send query to central directory → replies with list of matches
- Tradeoffs
  - How do you find the central directory service?
    - Typically using multicast based discovery!
    - SLP also allows directory to do periodic advertisements
  - Need dedicated infrastructure
  - How do directory agents interact with each other?
  - Well suited for browsing and composition → knows full list of services

Service Discovery (Routing Based)

- Client issues query to overlay network
  - Query can include both service description and actual request for service
- Overlay network routes query to desired service[s]
- If query only description, subsequent interactions can be outside overlay (early-binding)
- If query includes request, client can send subsequent queries via overlay (late-binding)
  - Subsequent requests may go to different services agents
  - Enables easy fail-over/mobility of service
- Tradeoffs
  - Routing on complex parameters can be difficult/expensive
  - Can work especially well in ad-hoc networks
  - Can late-binding really be used in many applications?

Wide Area Scaling

- How do we scale discovery to wide area?
  - Hierarchy?
- Hierarchy must be based on attribute of services
  - All services must have this attribute
  - All queries must include (implicitly or explicitly) this attribute
- Tradeoffs
  - What attribute? Administrative (like DNS)? Geographic? Network Topologic?
  - Should we have multiple hierarchies?
  - Do we really need hierarchy? Search engines seem to work fine!

Other Issues

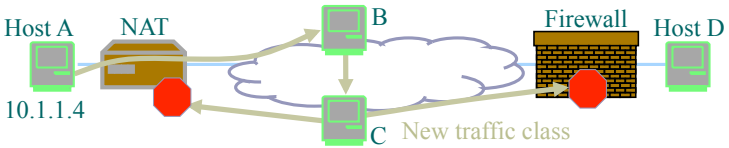
- Dynamic attributes
  - Many queries may be based on attributes such as load, queue length
  - E.g., print to the printer with shortest queue
- Security
  - Don't want others to serve/change queries
  - Also, don't want others to know about existence of services
    - Srini's home SLP server is advertising the $50,000 MP3 stereo system (come steal me!)

77

78

The Problem

- Middlebox: interposed entity doing more than IP forwarding (NAT, firewall, cache, …)
- Not in harmony with the Internet architecture



- No unique identifiers and on-path blocking:
  - Barrier to innovation
  - Workarounds add complexity

79

Reactions to the Problem

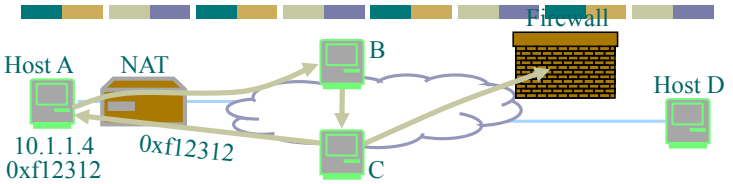
- Purist: can't live with middleboxes
- Pragmatist: can't live without middleboxes
- Pluralist (us): purist, pragmatist both right

Our goal: Architectural extension in which:

- Middleboxes first-class Internet citizens
- Harmful effects reduced, good effects kept
- New functions arise

80

DOA: Delegation-Oriented Architecture



Architectural extension to Internet. Core properties:

1. Restore globally unique identifiers for hosts
2. Let receivers, senders invoke (and revoke) off-path boxes: *delegation primitive*

Outline

I. DOA (Delegation-Oriented Architecture)

II. Uses of DOA

III. Related Work / Conclusion

Separate References and User-level Handles

tussle space
[Clark et al., 2002]

- "So aren't you just moving the problem?"
  - Yes.
  - But.

Let people fight over handles, not references