Information centric networking

Outline

- V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, R.L. Braynard. "Networking named content" , CoNEXT'09.
- S.K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsiz, T. Koponen, B.M. Maggs, K.C. Ng, V. Sekar, S. Shenker. "Less Pain, Most of the Gain: Incrementally Deployable ICN" , SIGCOMM'13.

Outline

- V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, R.L. Braynard. "Networking named content" , CoNEXT'09.
- S.K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsiz, T. Koponen, B.M. Maggs, K.C. Ng, V. Sekar, S. Shenker. "Less Pain, Most of the Gain: Incrementally Deployable ICN" , SIGCOMM'13.

Yesterday's and today's Internet

60's and 70's

- scarce resources; small number of hosts
- created for resource sharing (e.g., card reader)
- client/server (point-to-point)
- static; always connected
- slow links, slow CPUs, small memory; expensive
- best effort, no extra features for security, routing

Now

- increasing number of devices, some disposable
- mostly content sharing
- one produces, many consume; increasing amount of data
- increasing mobile, potentially intermittent connectivity
- ever fast links/CPU, abundant storage; cheap
- increased dependency, increased security threats

Claim: Internet communication model based on problems and technologies from 60's, 70's, which today is inadequate

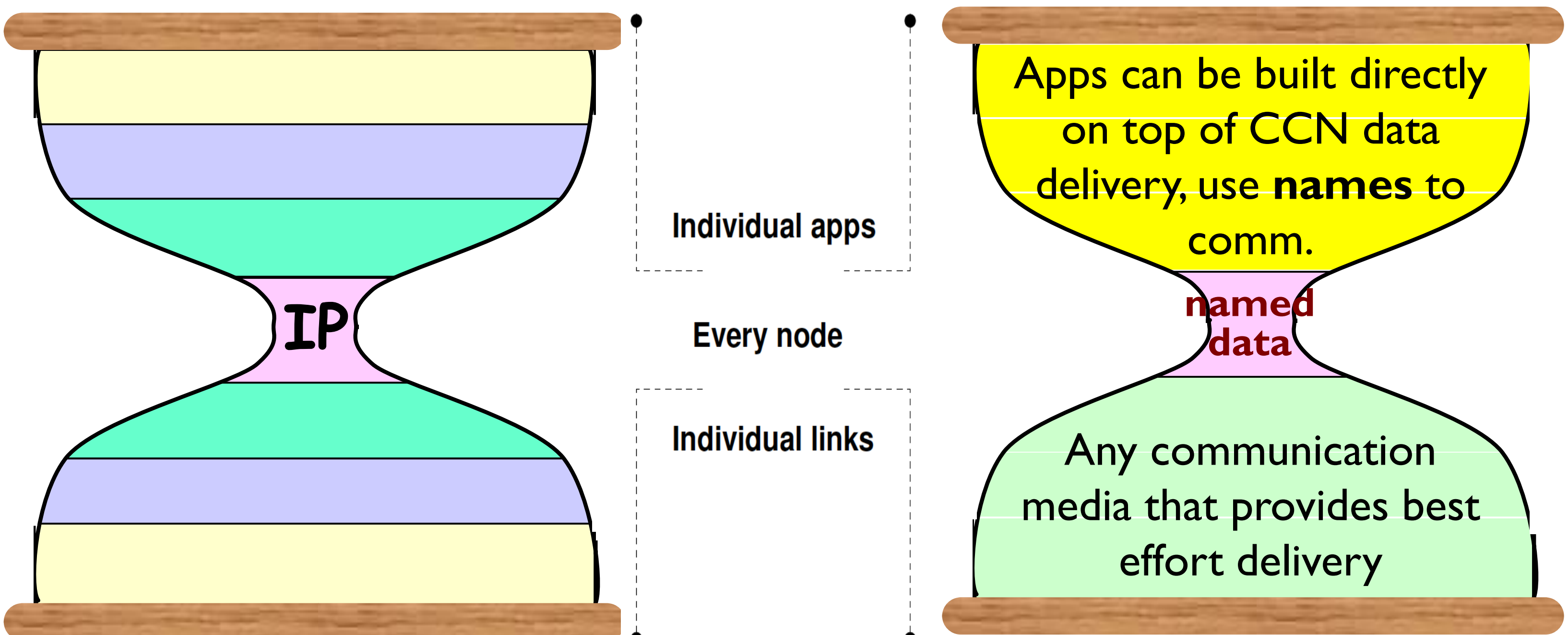
What needs to be changed?

- delivery to IP addresses
- point-to-point delivery
- lack of built-in security

How to make change?

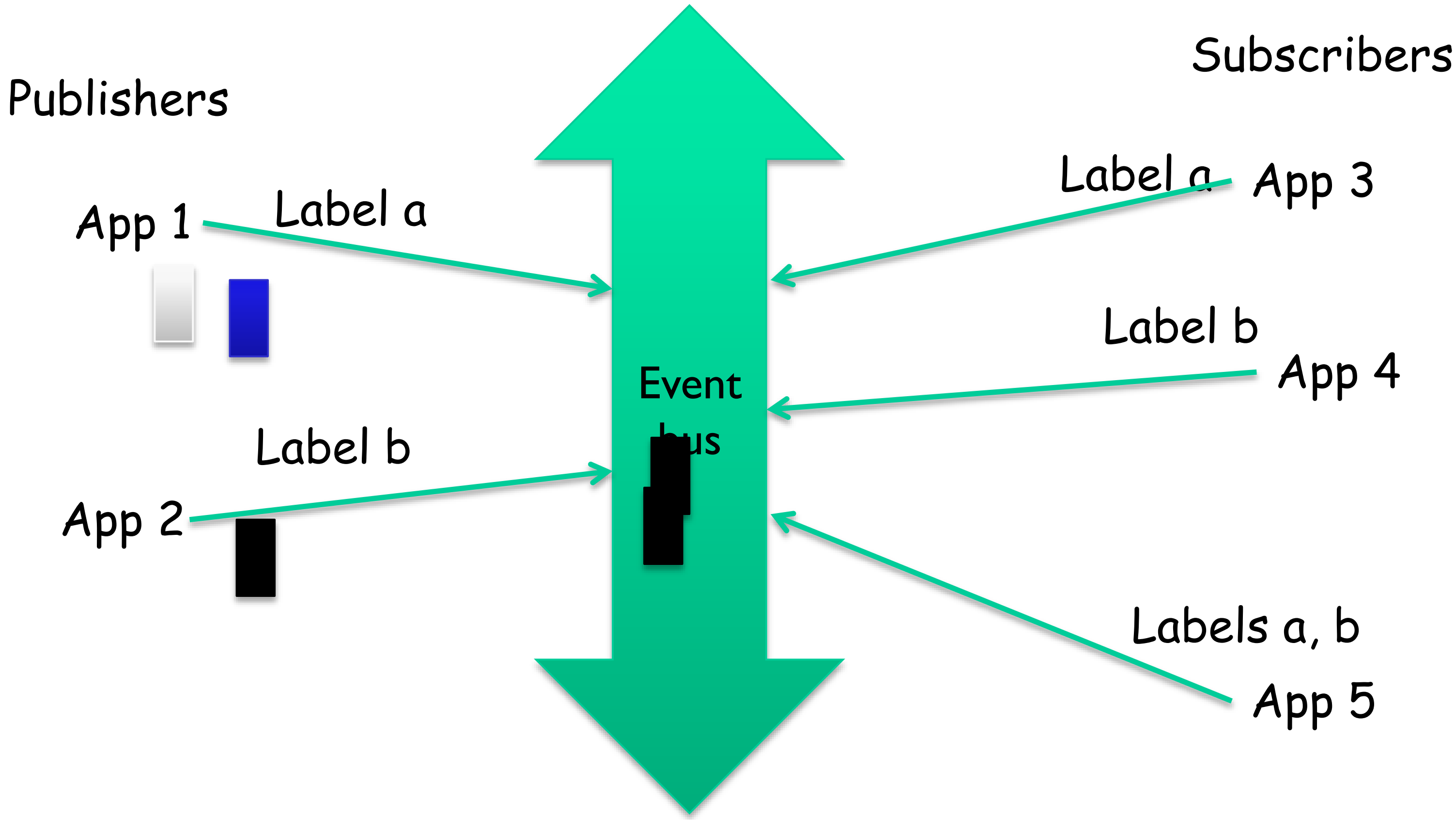
Content centric networks

Today's Internet delivers packets to destination IP addresses



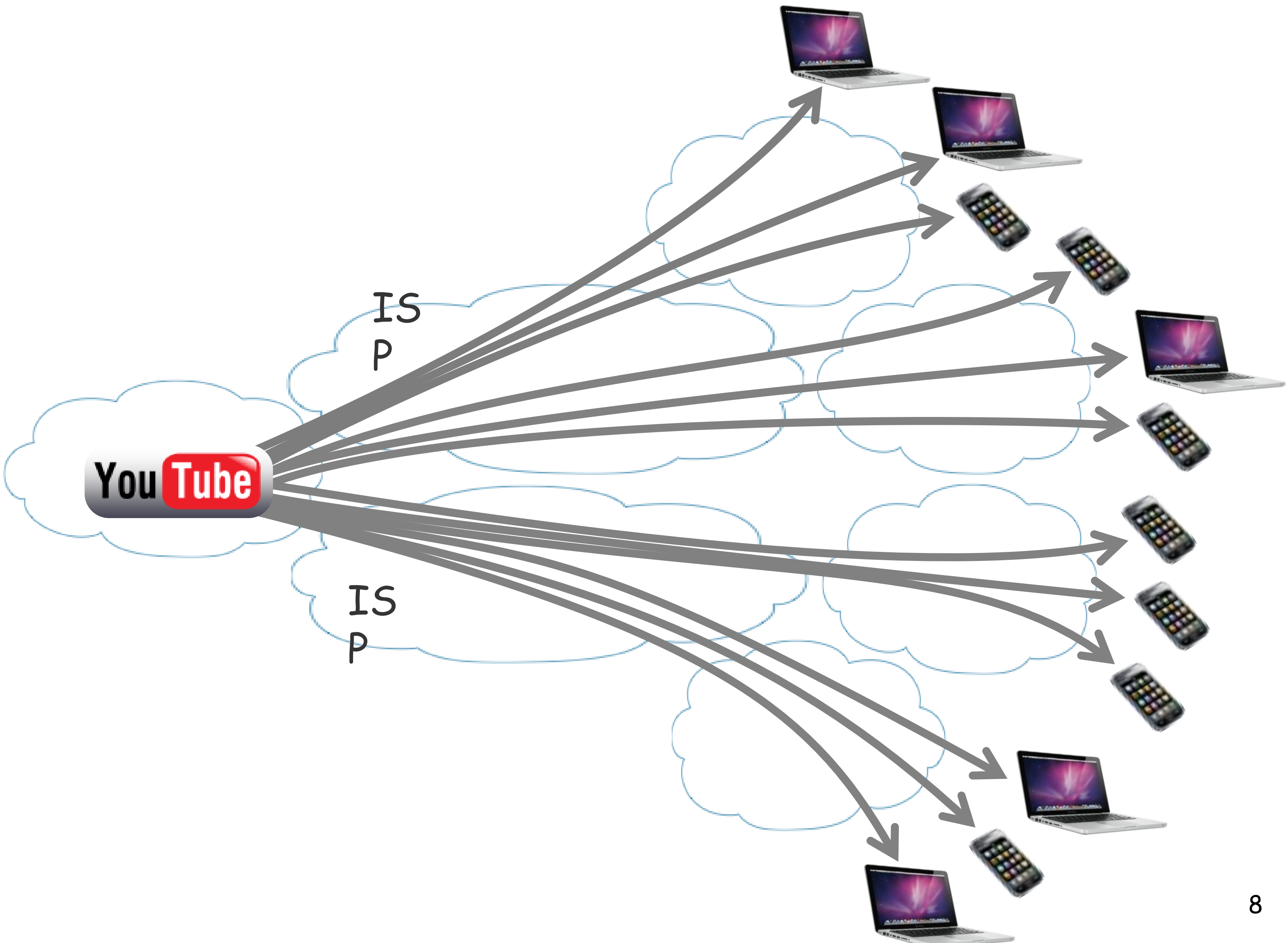
CCN moves <u>universal component</u> in Internet protocol stack from IP packets to named data

Digression: Publish/Subscribe Model

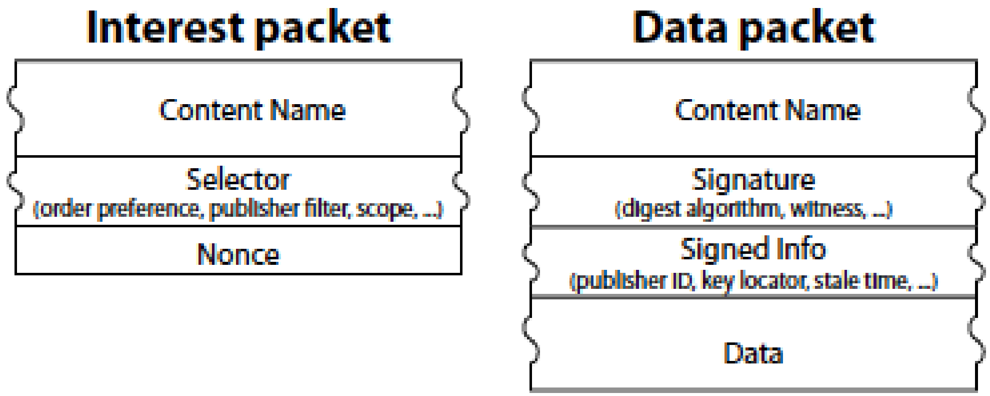


CCN requires content to be requested explicitly
Pub/Sub model has been proposed as enhancement

Problem with point to point model



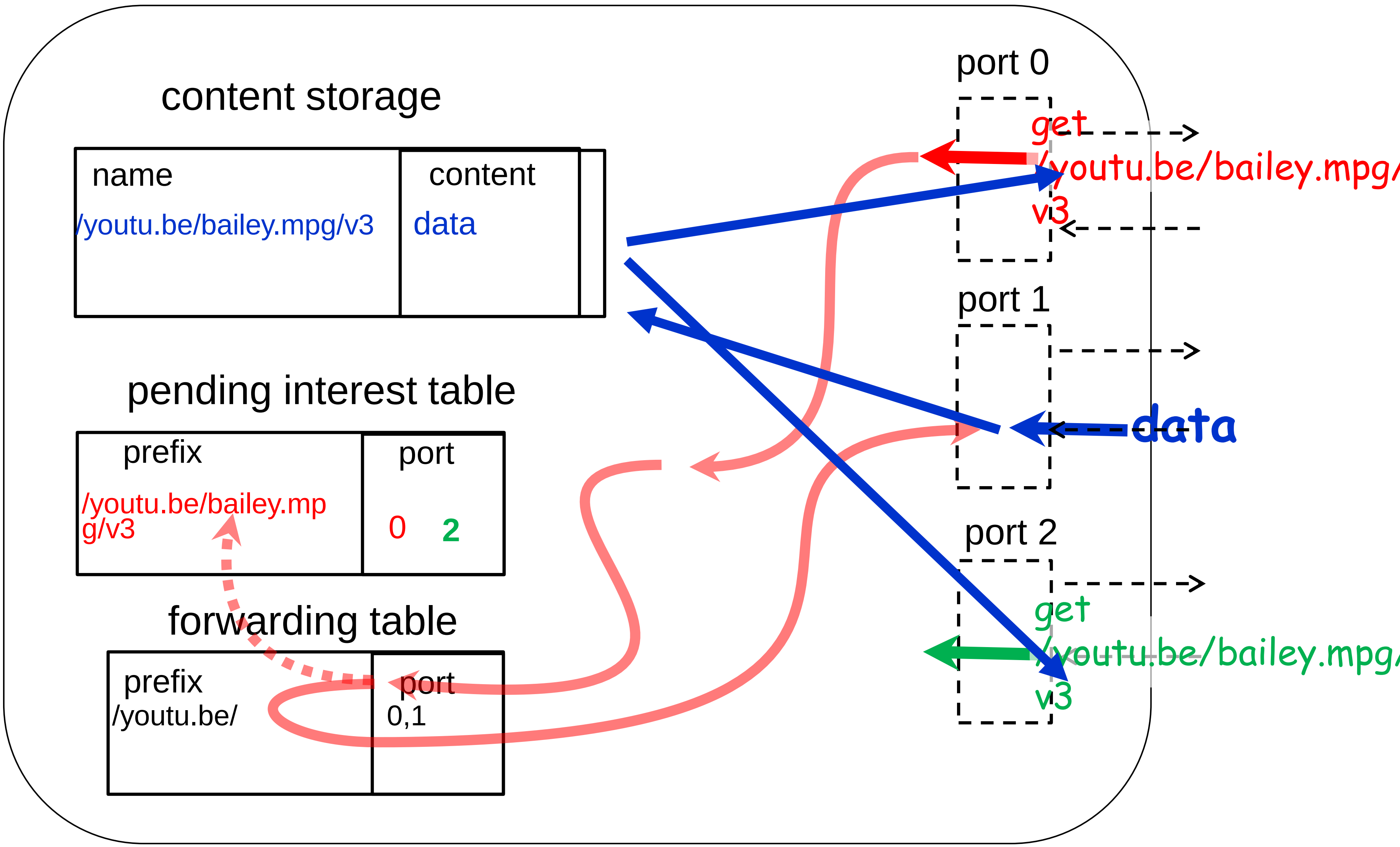
CCN architecture



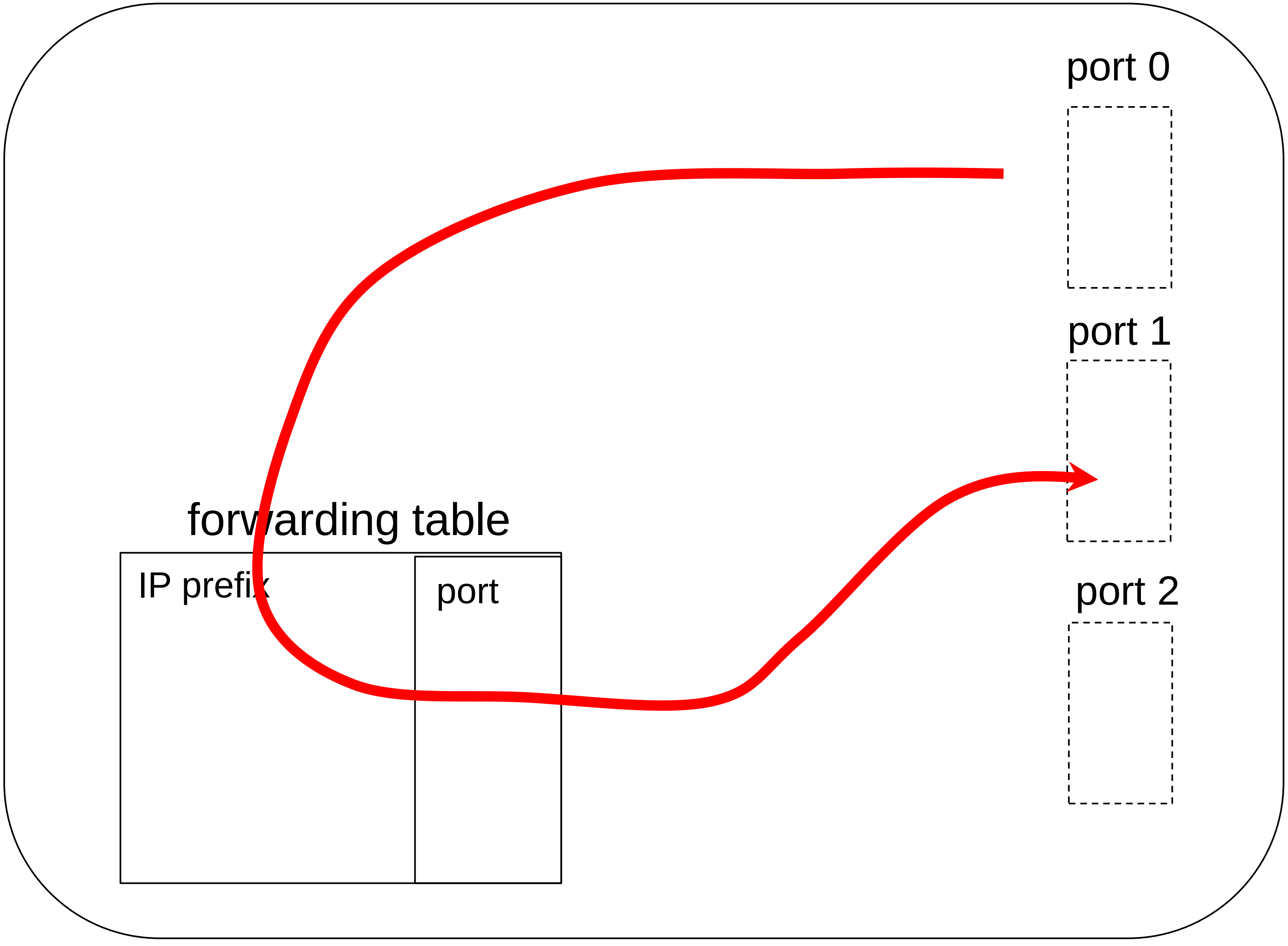
CCN Packets:

- consumers send **interest packets**, and nodes that can satisfy those interests respond with **data packets**
- **hierarchical** and **context-dependent** name prefixes (e.g., /local/Friends)
- **nonces** to prevent Interests from looping

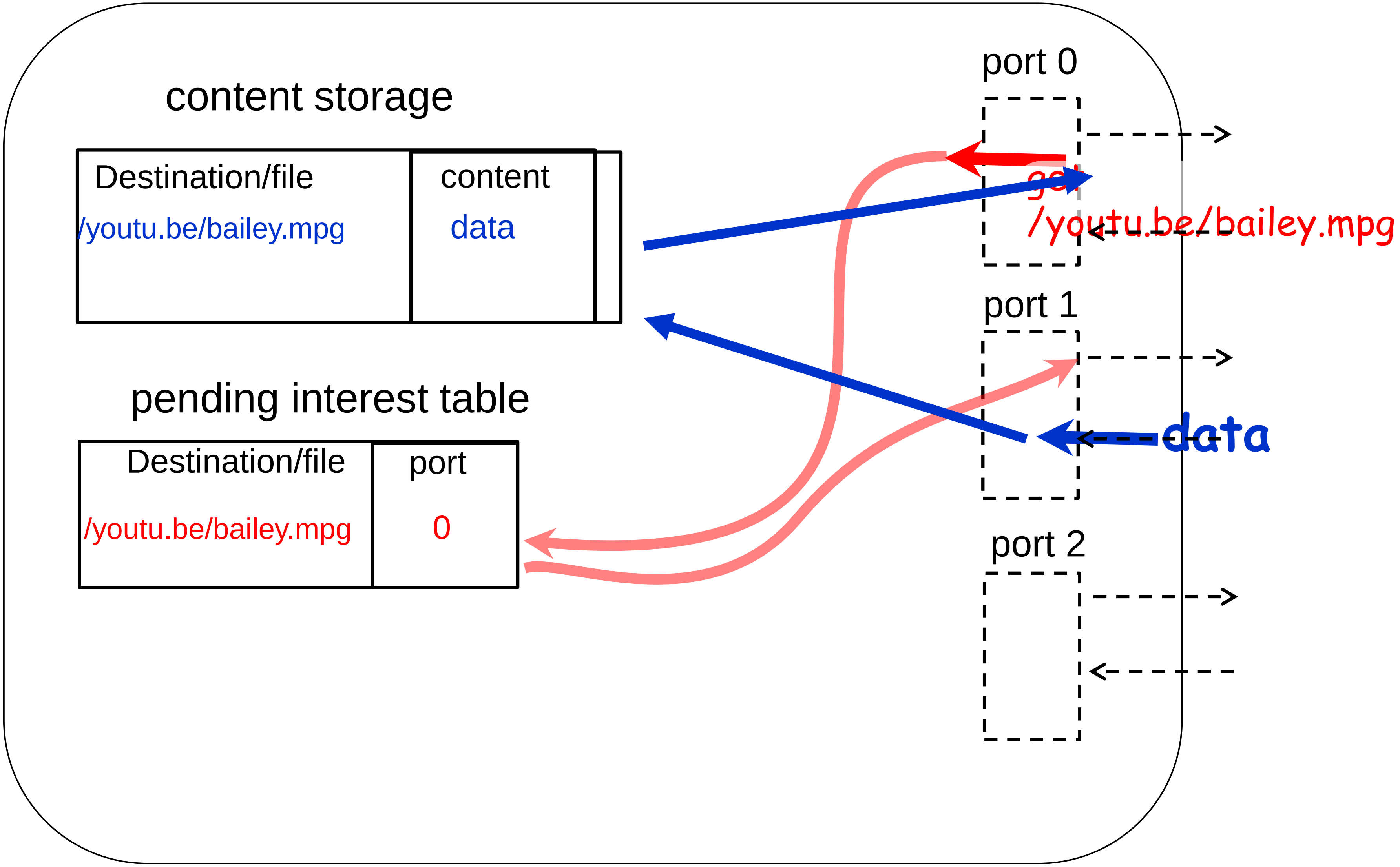
Routing functionality (Jacobson)



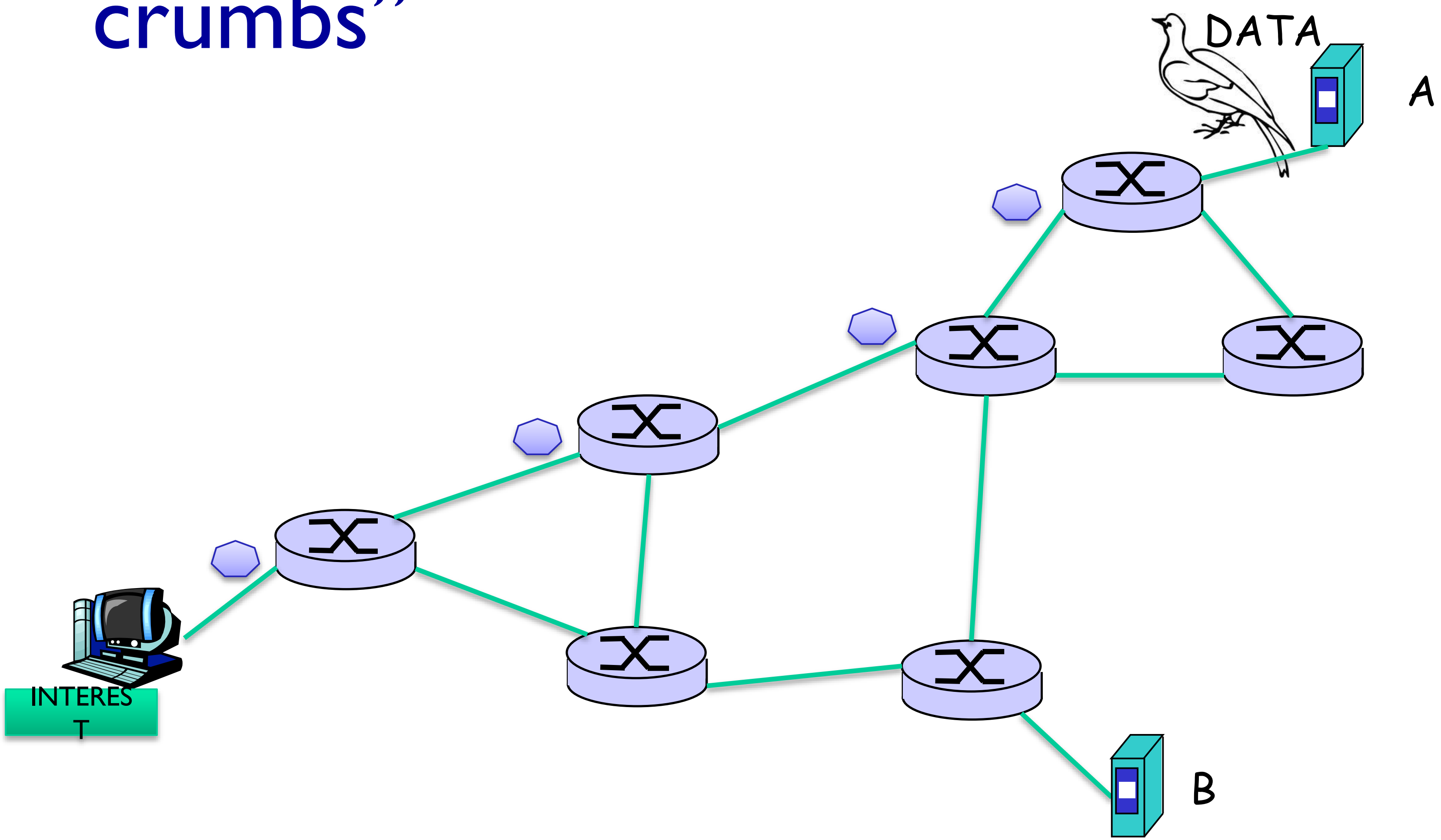
IP router



Web Cache (Proxy caching)



Pending interest table and "bread crumbs"

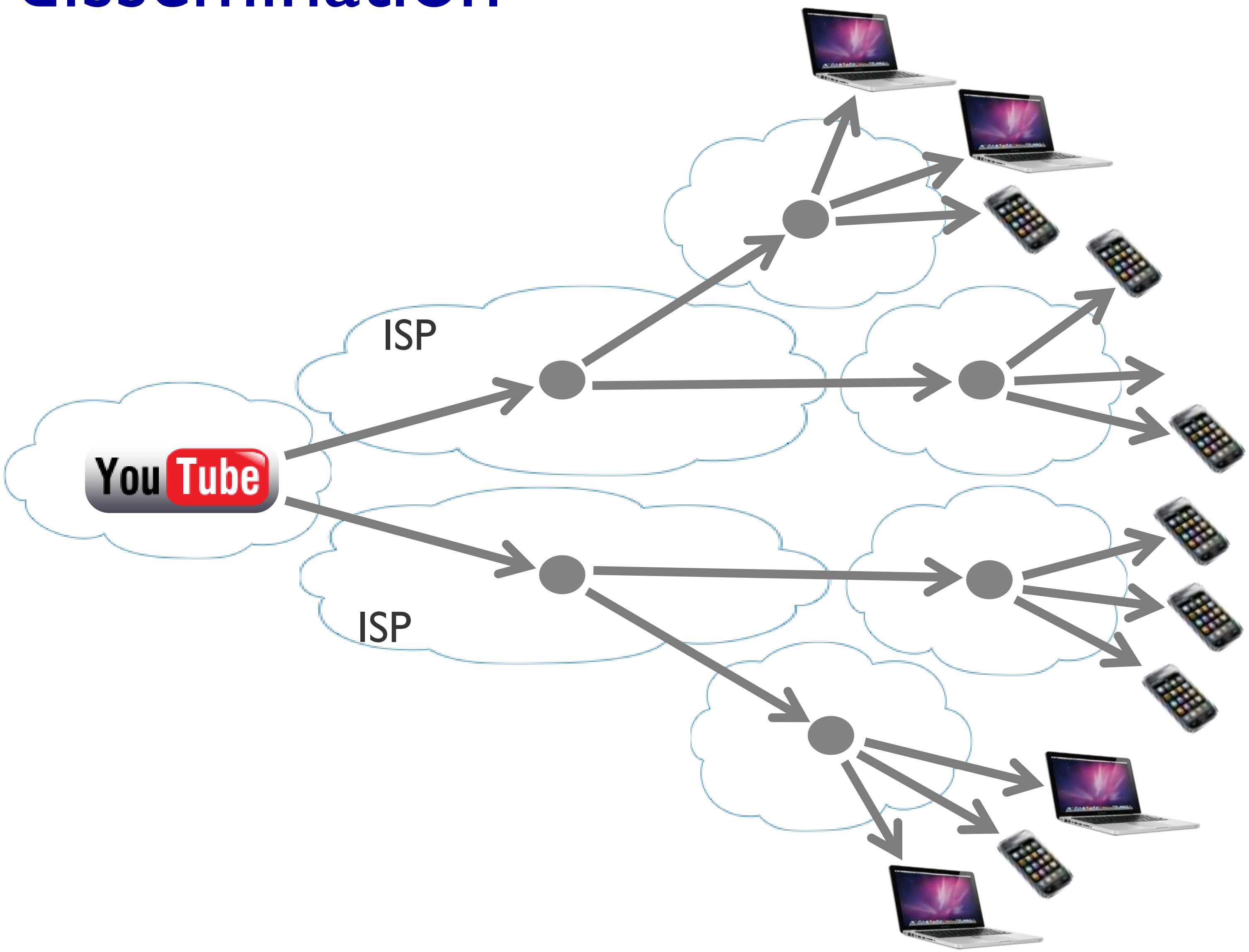


What happens if both A and B both have content?
If router sends interests on more than one face, first data pkt returned; some bandwidth wasted

Some remarks

- stateful routers
- interest routed, not data (breadcrumbs)
- duplicate data packets are discarded
- **nonces** (random numbers) prevent Interest packets from looping
- content store uses favorite replacement scheme
- pending interest entries have timeouts

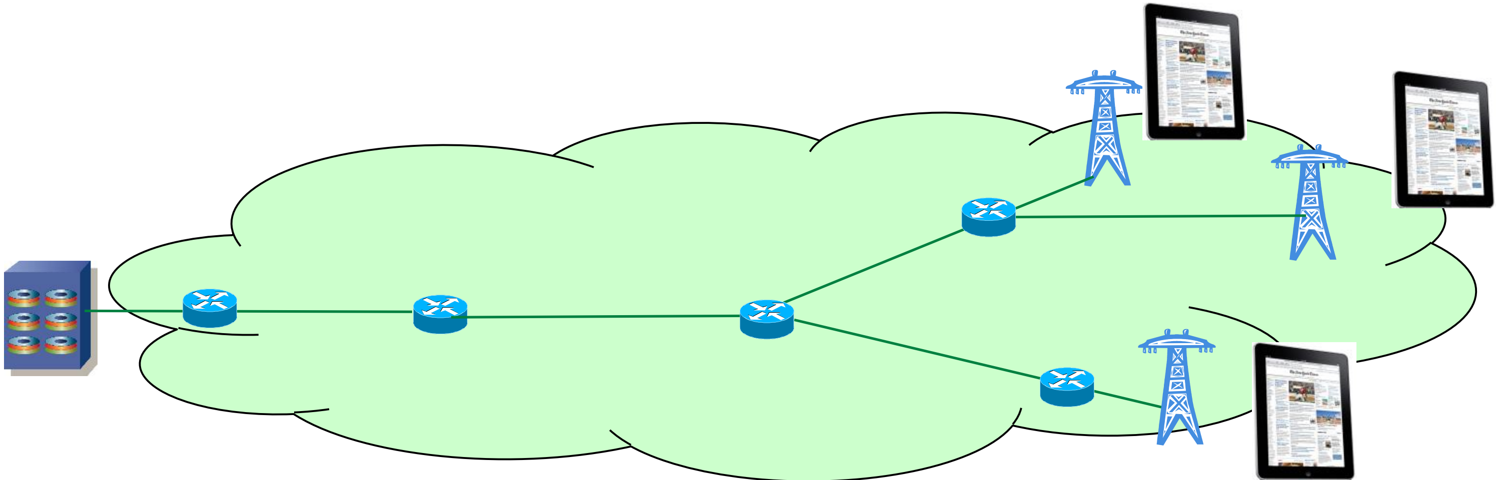
CCN enables scalable data dissemination



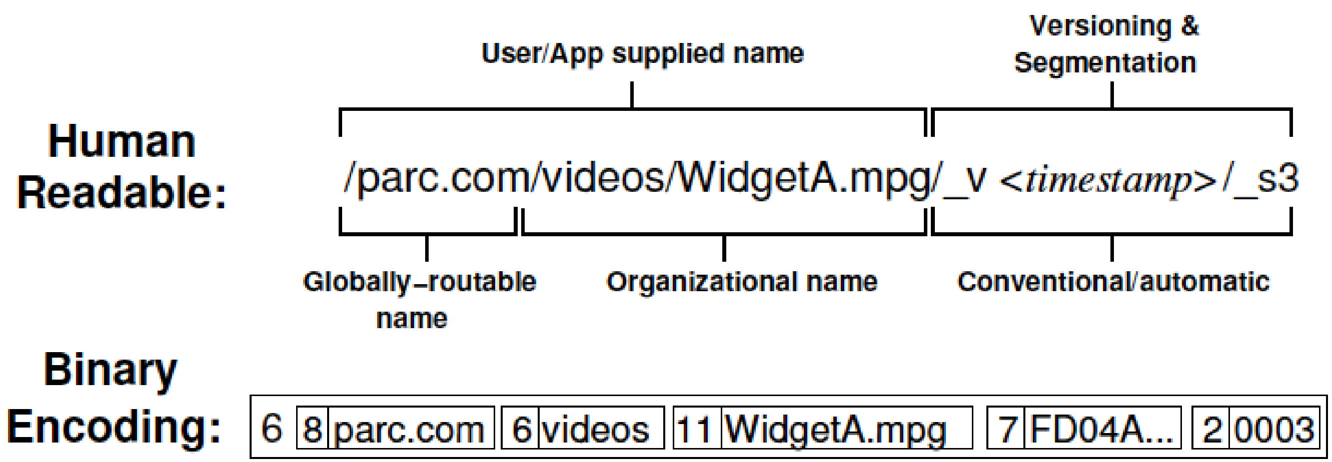
Ad hoc networking, mobility, DTN

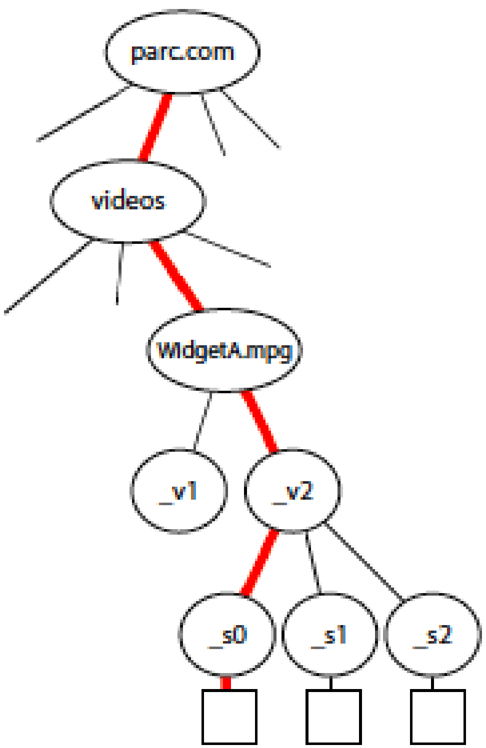
- two or more mobile nodes can start communicating with each other as soon as they can physically reach each other

- CCN provides efficient streaming to mobiles on the move



Addressing Scheme





- hierarchical names (components)
- sequencing

What is maximum height/width of tree?
How long does look-up take?

TCP-like features

Reliability:

- application resends requests, more flexible
  E.g., app can implement network coding
- similar to TCP SACK

Flow Control:

- at most one data packet per interest packet
- TCP window advertisements → interest packets

Other layers

Strategy layer (program in Forwarding Table describing how to use faces)

- multiple interfaces allowed
  - sendToAll (broadcast), sendToBest (opportunistic routing)

Routing:

- any routing scheme that works well for IP
  - IP and CCN forwarding are based on prefixes
- multi-sources, multi-destinations
- compatible with IP-based routers (CCN route announcements discarded)

CCN Security Model

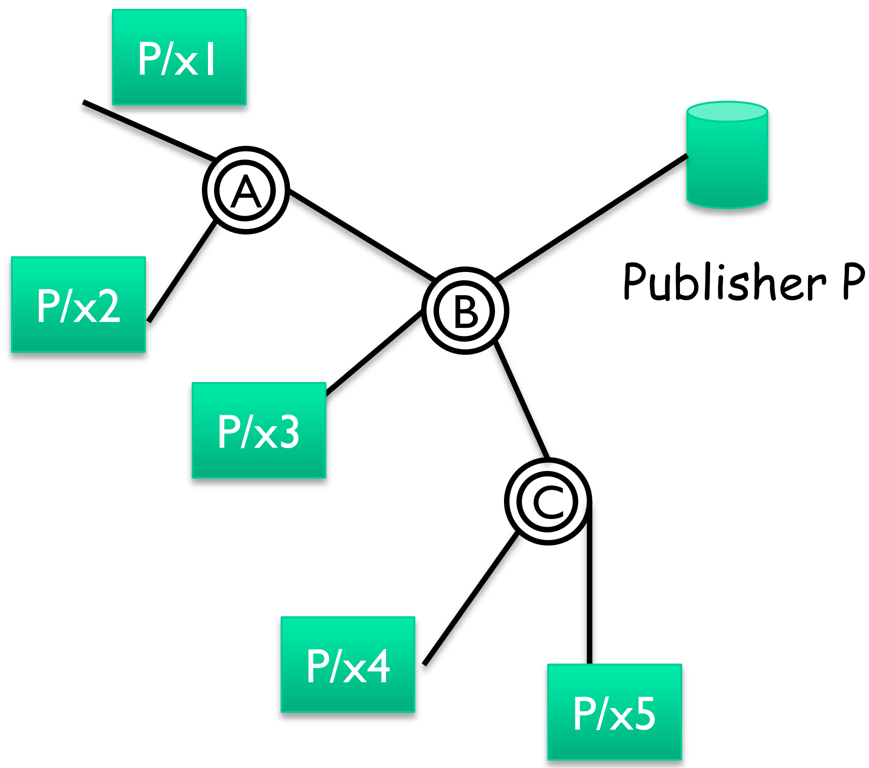
- IP: point-to-point, secures the channel
- CCN: secures data, not its container
  - first, data must be visible in the architecture
  - then, secure data:
    - associate key with each name, sign data together with name at creation
    - can verify integrity and provenance independent from where it came

Interest flooding attack

Flooding: *Generate large number of interest packets to overwhelm content source*

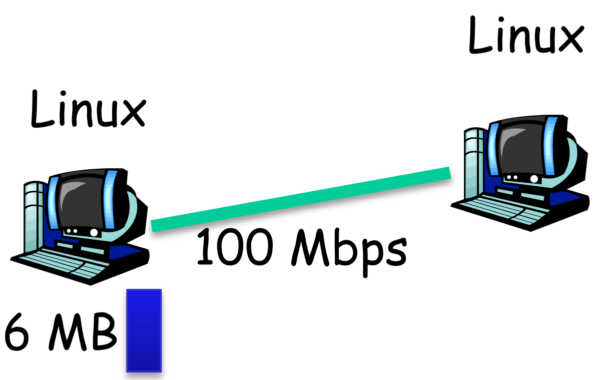
Defenses:

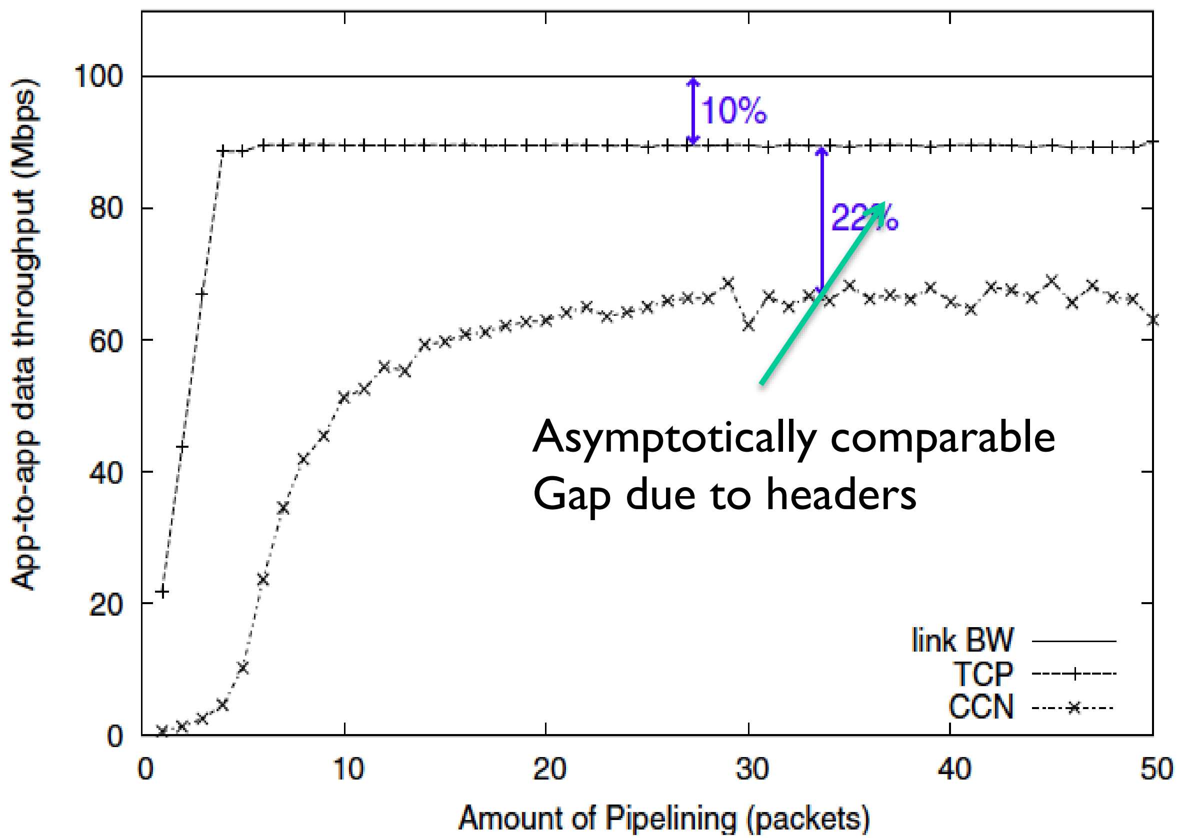
- nodes can monitor how many interest pkts of same prefix were successfully resolved
- domain can ask **downstream routers** to throttle number of interests they forward of same prefix



Experiment 1

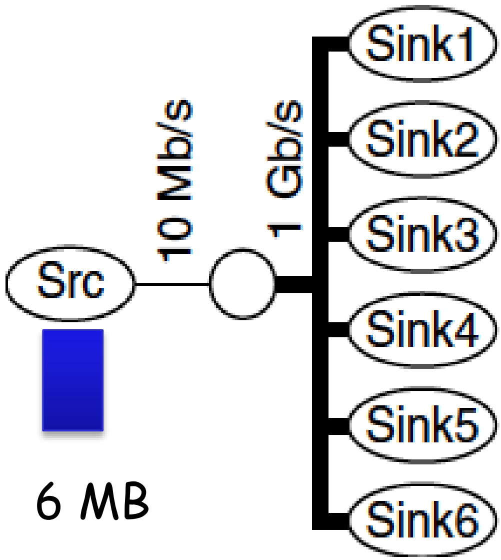
- performance of CCN vs. tcp

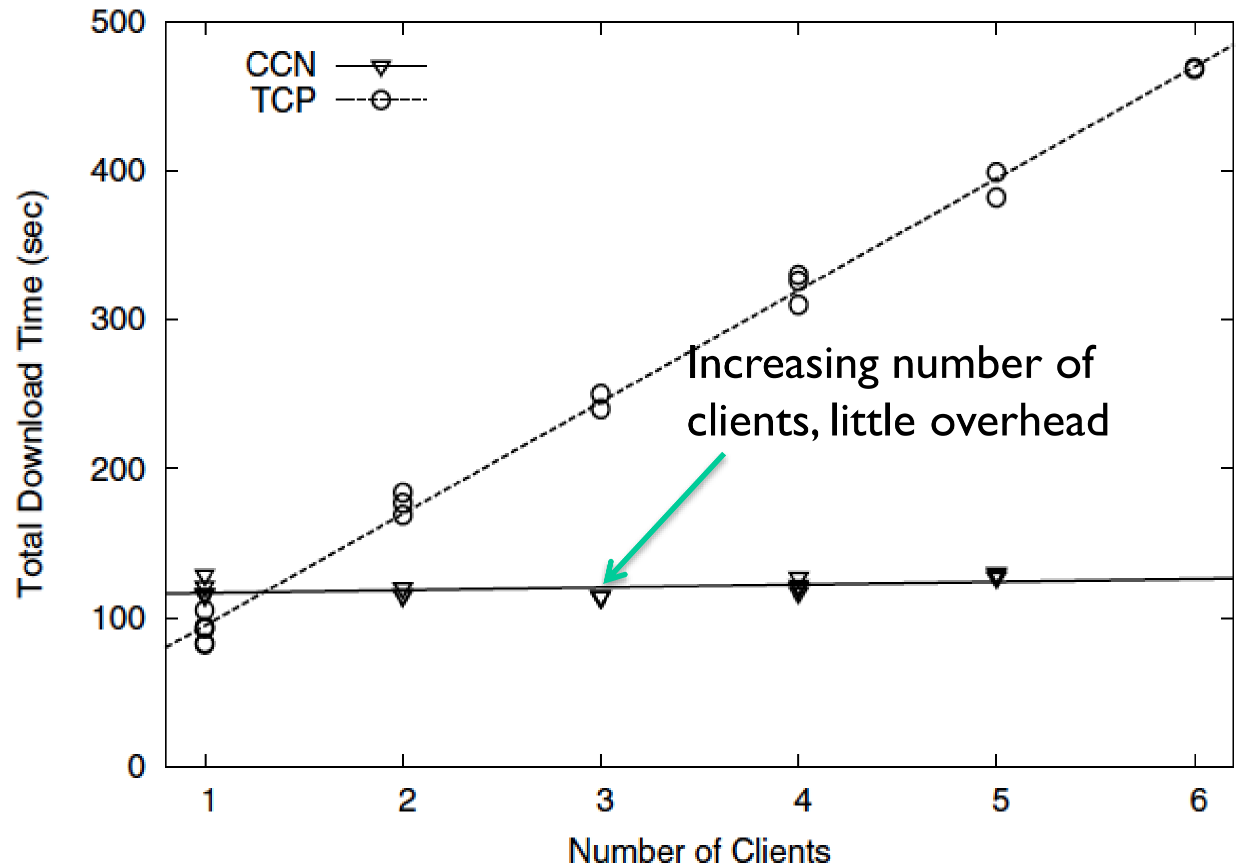




Asymptotically comparable
Gap due to headers

Experiment 2: “multicast”performance





Implicit assumption - Sinks sync’d

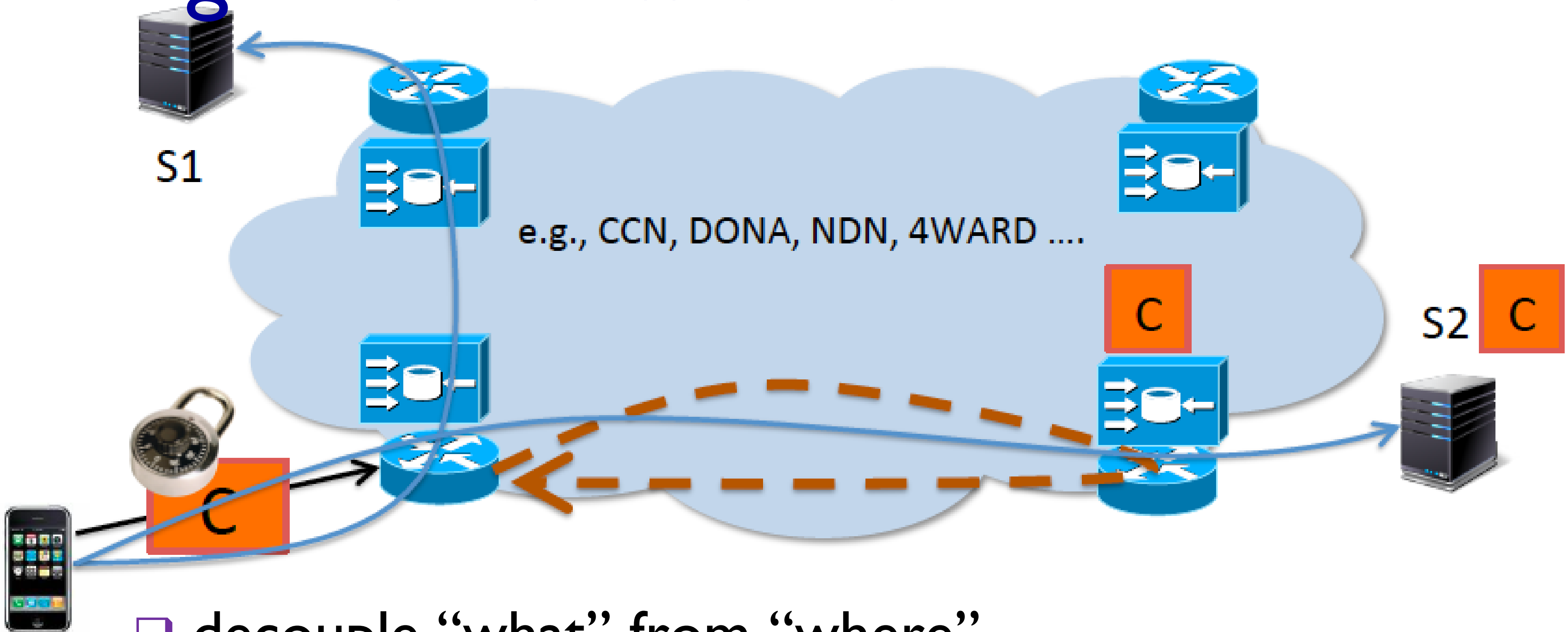
Summary

- CCN - clean-slate architecture for content-based network service
- based on successes and lessons from today's Internet
- built-in security, multicast and multipath
- components to facilitate mobility, ad hoc and disruption-tolerant networks
- incrementally deployable, but nodes in "bridged" CCN-capable ISPs won't see benefits
- supports consumer mobility
  - provider mobility?

Outline

- V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, R.L. Braynard. "Networking named content" , CoNEXT'09.
- S.K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsiz, T. Koponen, B.M. Maggs, K.C. Ng, V. Sekar, S. Shenker. "Less Pain, Most of the Gain: Incrementally Deployable ICN" , SIGCOMM'13.

High-level view of ICN



- decouple "what" from "where"
- bind content names to content
- equip network with content caches
- route based on content names
  e.g.: find nearest replica

Motivation for work

Gains

- lower latency
- reduced congestion
- support for mobility
- intrinsic security

Can we achieve ICN gains without pains?

e.g., existing technologies?

e.g., incrementally deployable?

Pains

- routers need to be upgraded with caches
- routing needs to be content based

Approach: Attribute gains to architectural features

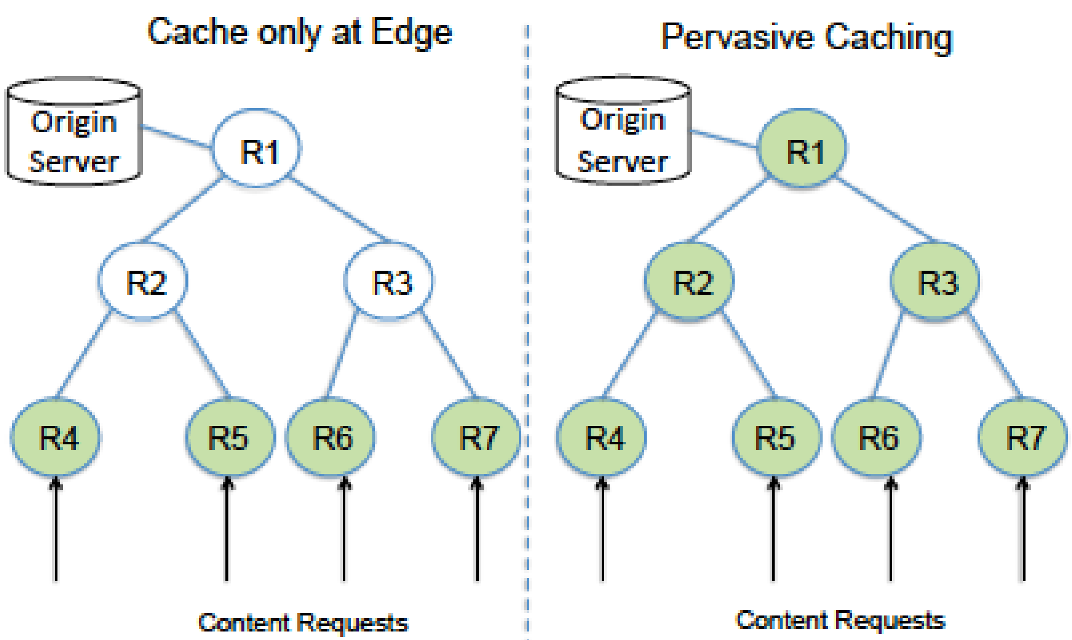
Quantitative **Qualitative**

- lower latency
- reduced congestion
- support for mobility
- intrinsic security

- decouple "what" from "where"
- bind content names to intent
- equip network with content caches
- route based on content names

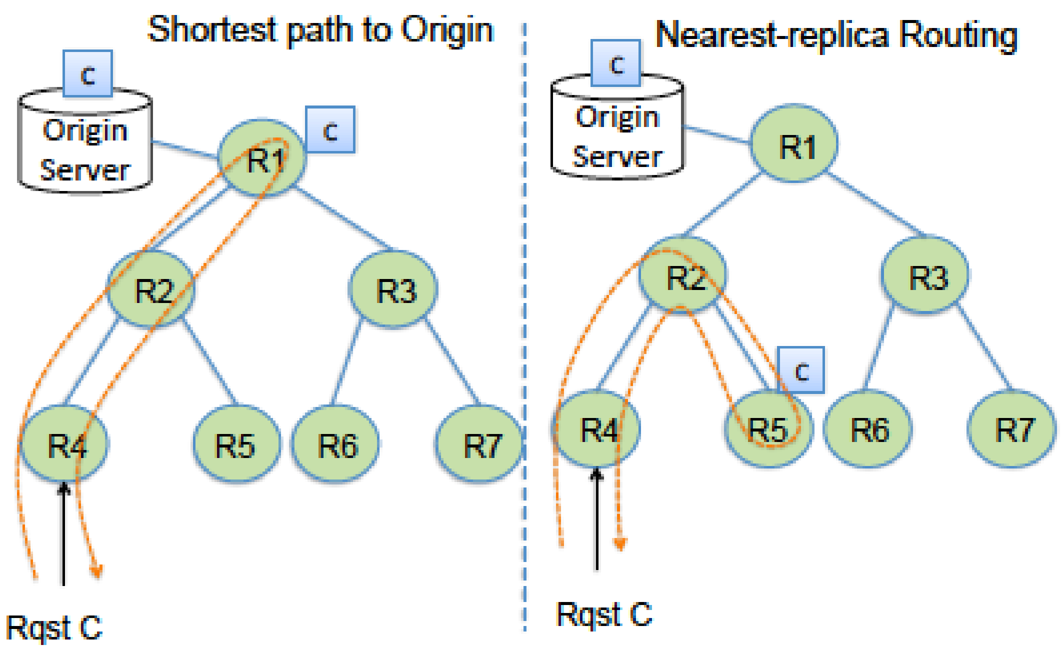
Representative designs

- *take-away*: **Improvements** on unicast transmissions largely **due to caching**
- two key dimensions to *design space*:
  1) cache placement: **Edge** vs. **Pervasive** (everywhere)



Representative designs

2) How to route requests
Shortest path to origin vs. **Nearest replica**



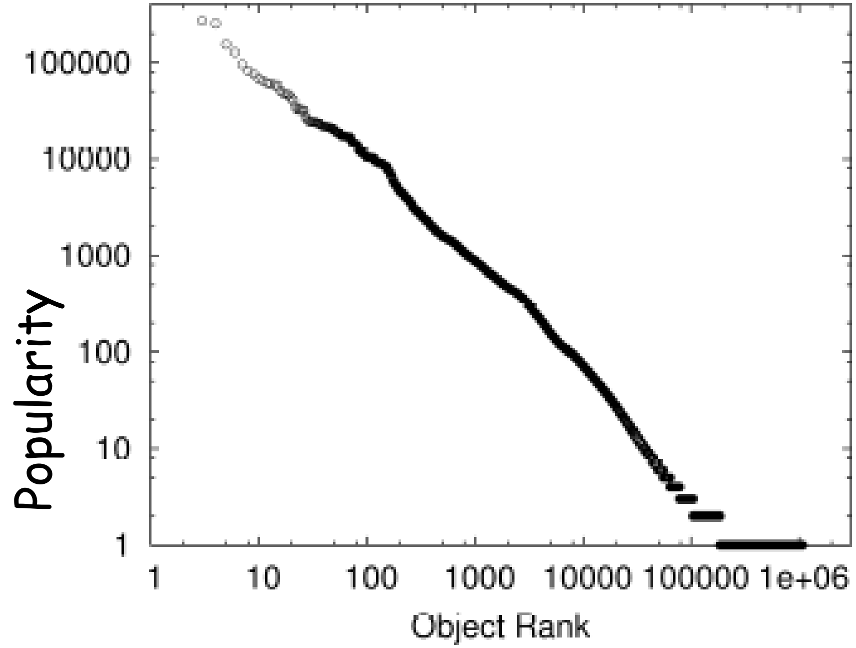
How is CCN (previous paper) classified?
Pervasive caching and nearest replica routing (?)

Heavy-tailed workloads

Heavy-tailed: informally, values much larger than average happen significantly often

X_k - popularity of *k*-th most popular file.

Zipf's law: $X_k \sim \frac{1}{k^{\alpha}}, \quad \alpha > 0$



(a) US

Key takeaways

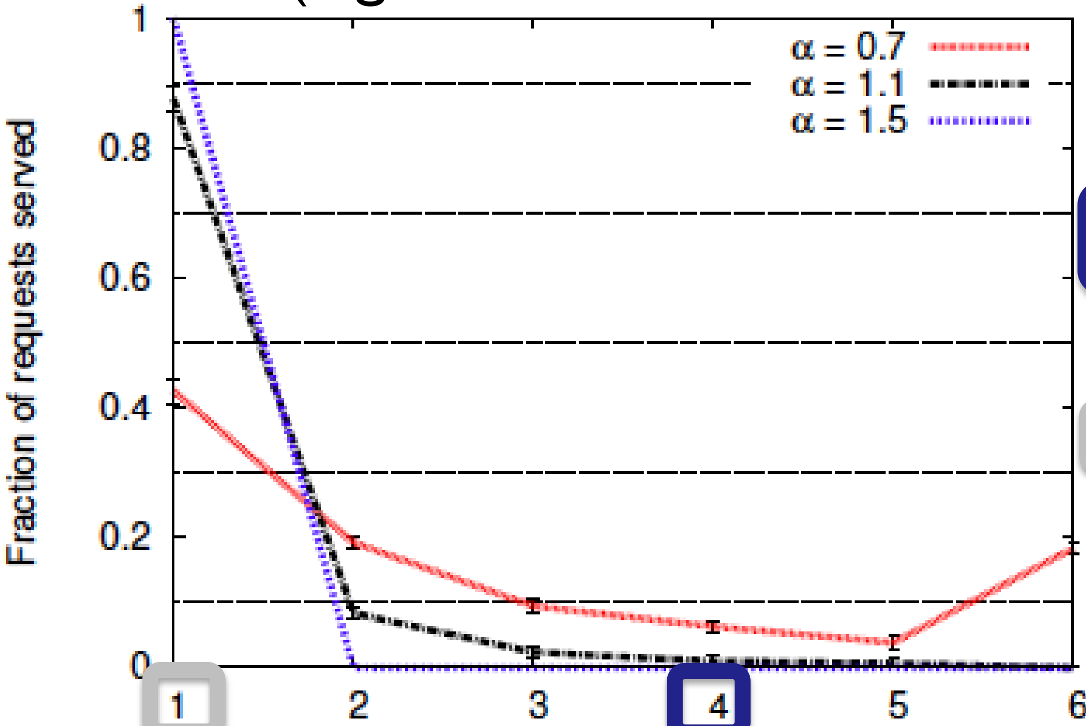
- to achieve quantitative benefits:
  → cache at "edge"
  → with Zipf-like workloads, pervasive caching and nearest-replica routing don't add much

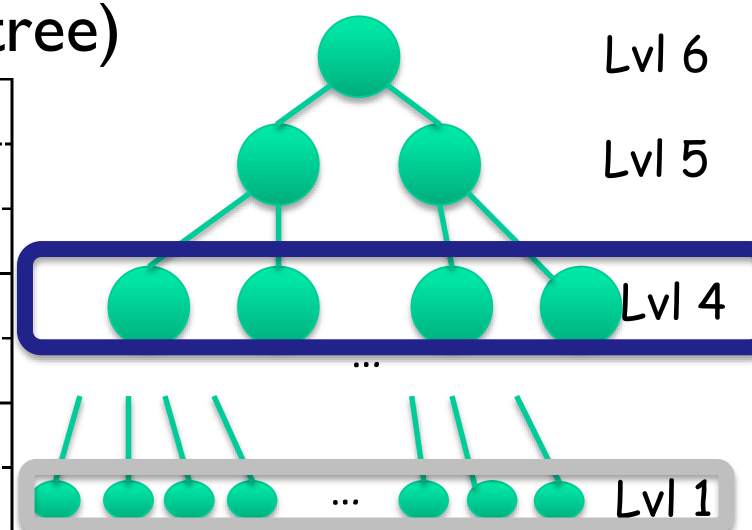
- to achieve qualitative benefits:
  → build on HTTP

Basis for incrementally deployable ICN

Heavy-tailed workloads: implications

- caching a few of most popular items yields large hit ratios
- larger exponent α, faster popularity decays
- decreasing improvement from setting extra nodes as caches (e.g., interior nodes of the tree)

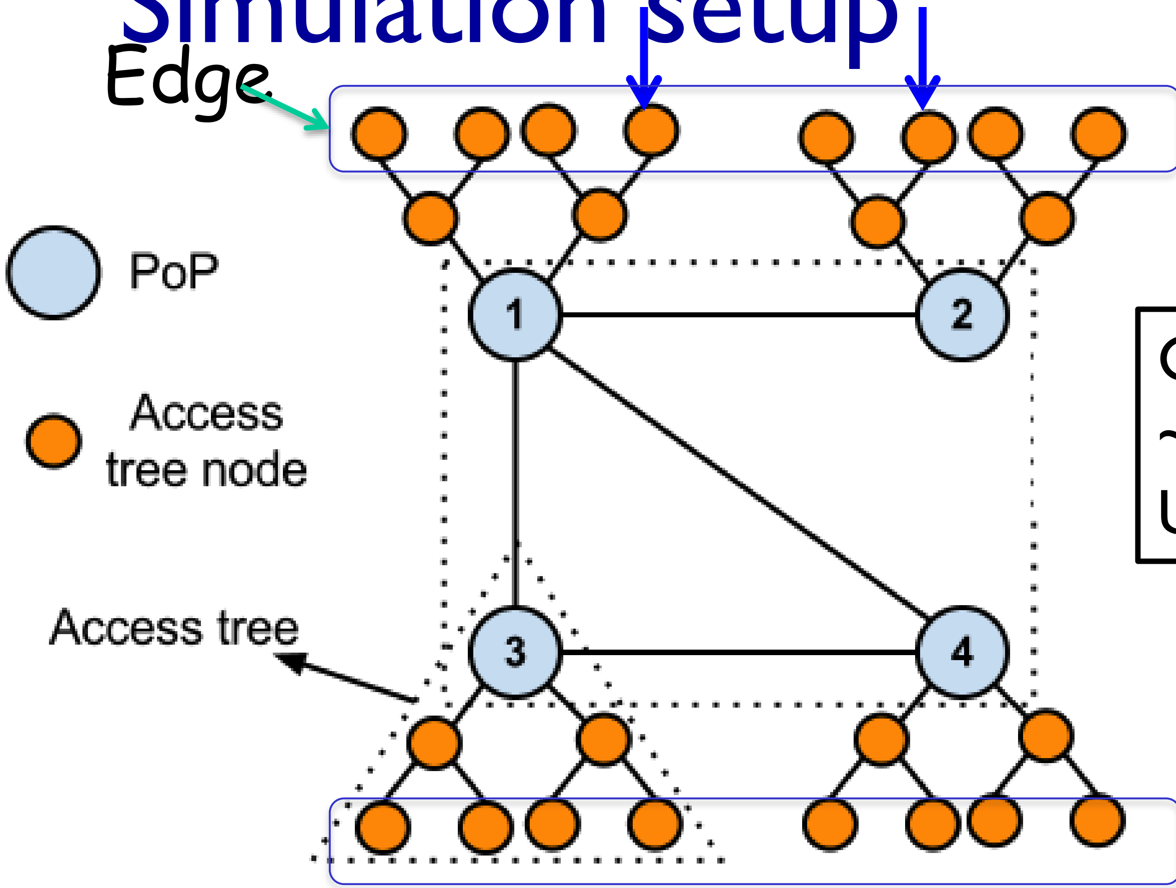




Location	Requests	Zipf parameter
US	1.1M	0.99
Europe	3.1M	0.92
Asia	1.8M	1.04

Take-away: caching at edges suffices

Simulation setup



Real CDN request logs

Cache provisioning ~ 5% of objects
Uniform requests

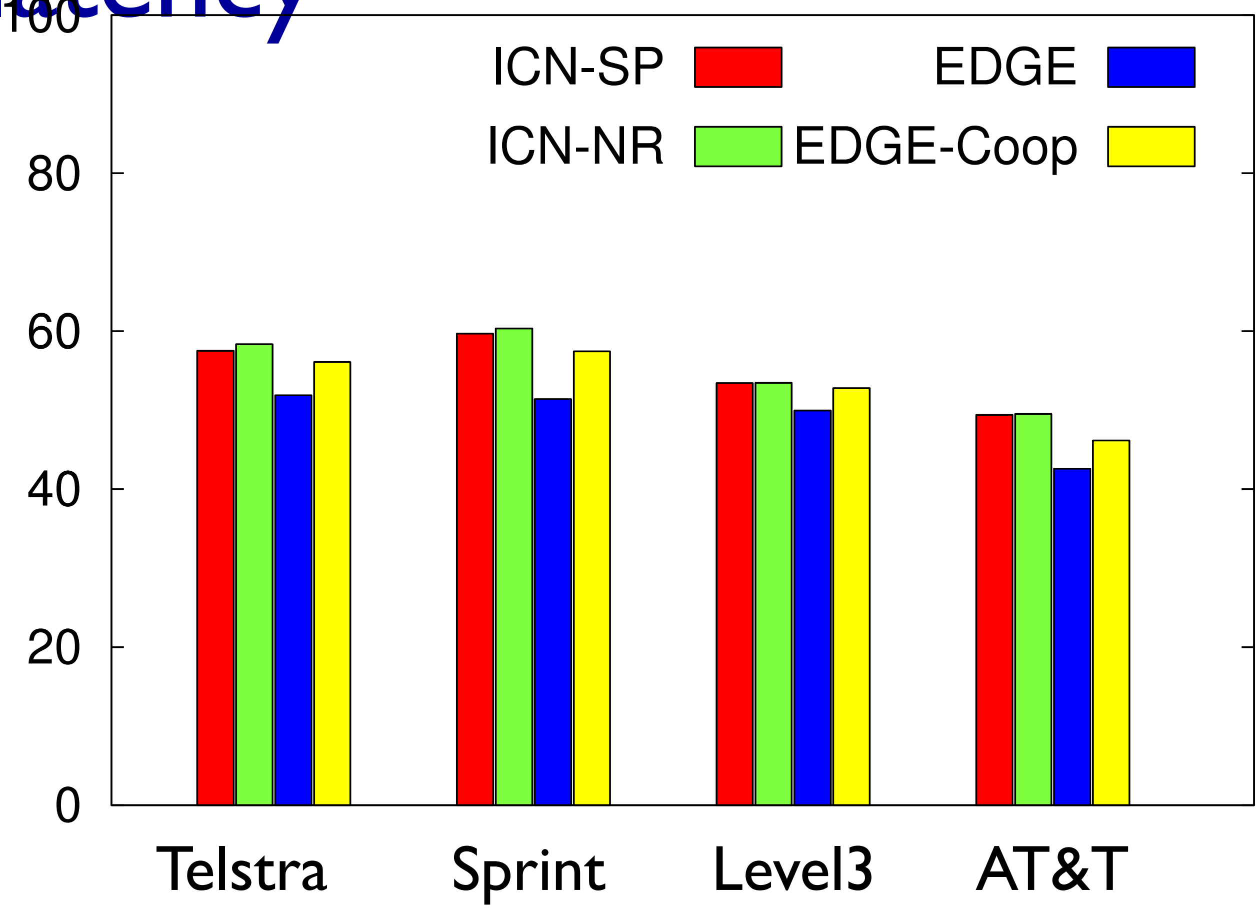
LRU replacement

PoP-level topologies (Rocketfuel) augmented with access trees

Assume name-based routing, lookup incurs zero cost

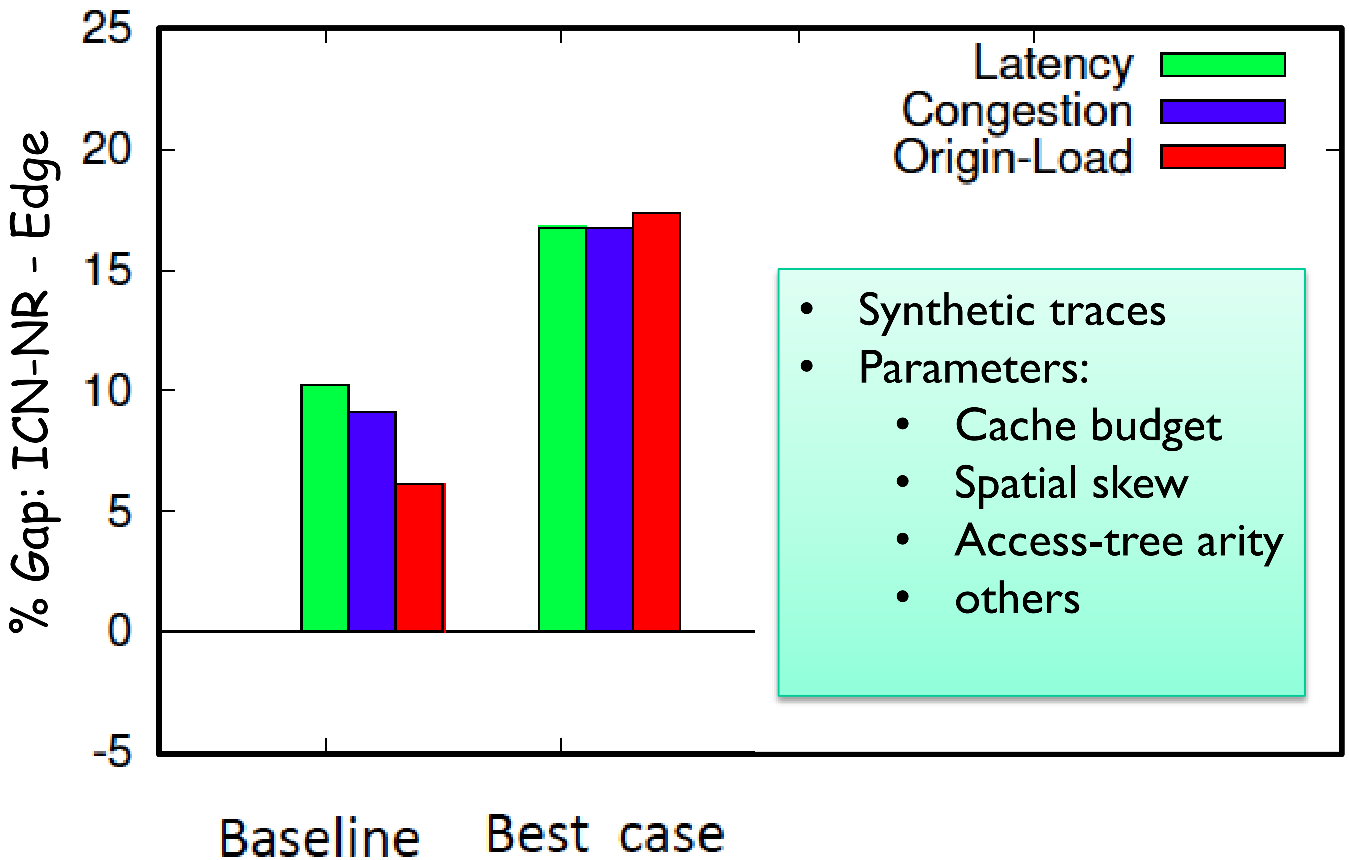
Request latency

% improvement over "no-cache"



Gap between architectures small (< 10%)
Similar results for congestion + server load

Sensitivity Analysis



- Synthetic traces
- Parameters:
  - Cache budget
  - Spatial skew
  - Access-tree arity
  - others

Little difference; in best case, ICN-NR only 17% better
Gap can be "easily" reduced
E.g., normalized budget or cooperative strategies

Implications of Edge Caching

- incrementally deployable
  - domains get benefits without relying on others
- incentive deployable
  - domains' users get benefits if domain deploys caches

Revisiting Qualitative Aspects

1. Decouple names from locations

Build on HTTP

– Can be viewed as providing "get-by-name" abstraction

– Can reuse existing web protocols (e.g., proxy discovery)

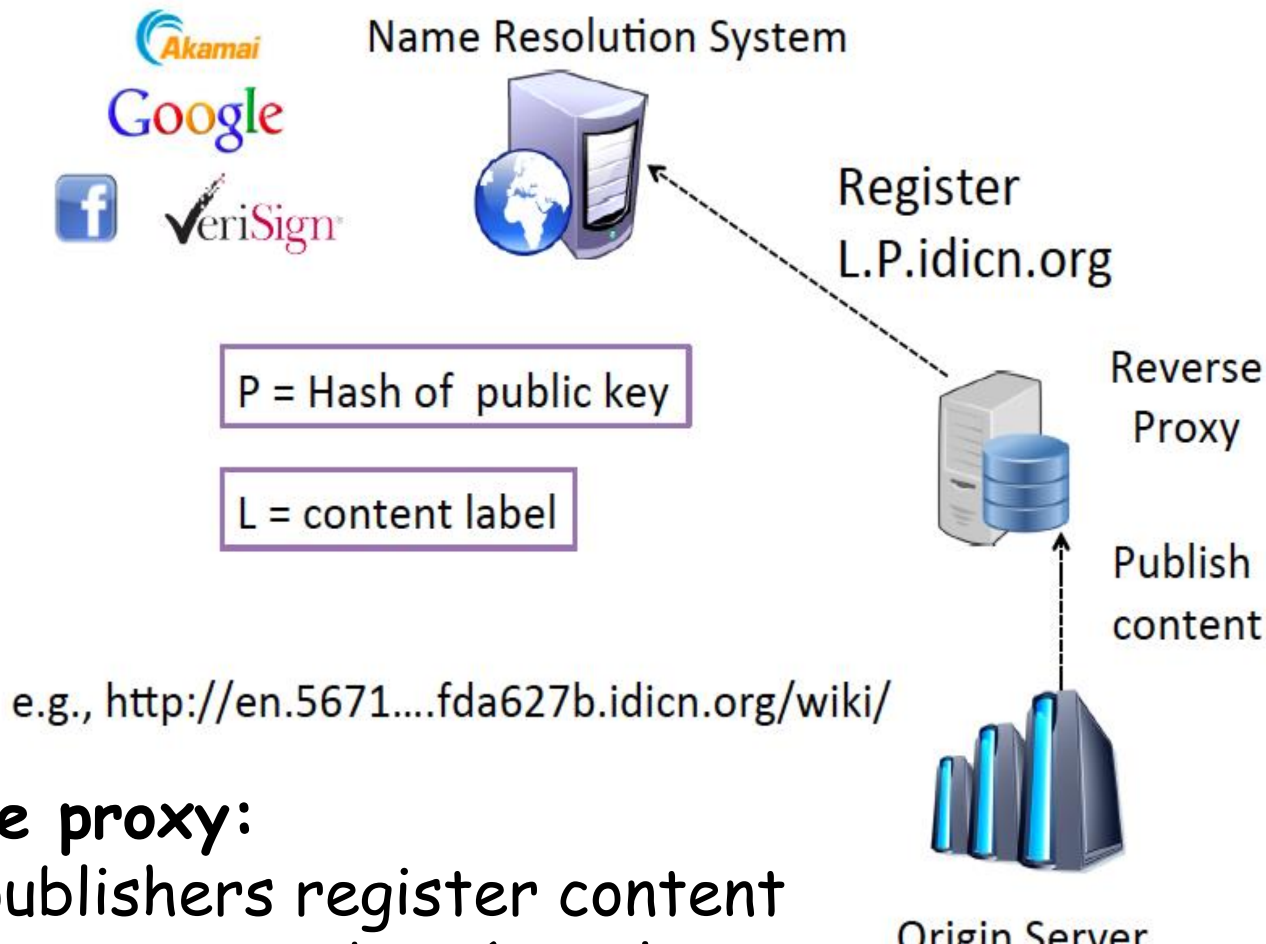
2. Binding names to intents

Use self-certifying names

e.g., "Magnet" URI schemes

Extend HTTP for "crypto" and other metadata

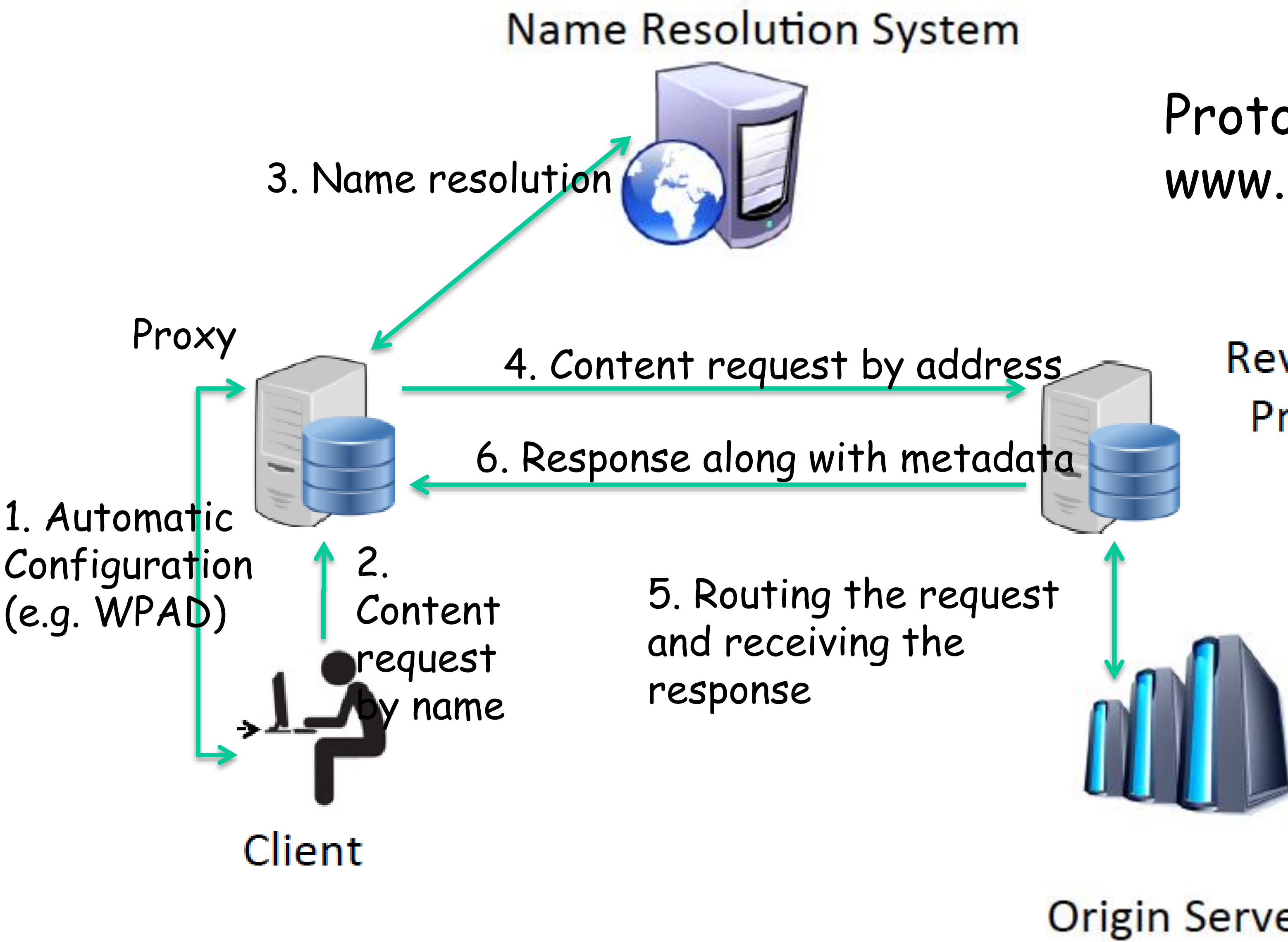
idICN: Content Registration



Reverse proxy:

- Let publishers register content
- Generates metadata (e.g., locations, signatures, policies)
- Receives requests by name and return content + metadata

idICN: Content Delivery



Prototype: www.idicn.org

Summary

- gains of ICN with less pain
  - latency, congestion, security
  - without changes to routers or routing
- quantitative benefits with “edge” solutions
  - **pervasive caching, nearest-replica** routing **not needed**
- qualitative benefits with existing techniques
  - existing HTTP + HTTP-based extensions
  - incrementally deployable
- idICN: one possible feasible realization
  - open issues: economics, other benefits, future workloads
- no multicast, support for mobility

Quick comparison

CCN

- clean-slate
- requires changing routers
- pervasive caching, nearest replica routing
- multiple source-destinations
- built-in security; protection against DoS attacks

idICN

- based on existing infrastructure/protocols
- edge caching, cooperative routing requests
- point-to-point
- security thru extending HTTP to negotiate metadata and standardizing self-certifying naming scheme

Both cases, produce networks of caches.
Evaluation involves understanding interaction between caches