NDN AND IP ROUTING CAN IT SCALE?

ASHOK NARAYANAN
DAVID ORAN

CISCO

ICN and content routing

- ICN models propose different mechanisms to route content requests towards content
  - Publish/Subscribe: Use IP routing underlay to establish an overlay graph of nodes that exchange control information about content and run TCP/HTTP-type transport for data
  - Content routing: Push content awareness down to routers; collapse the content overlay and the IP underlay
- These are really different points on a routing continuum, rather than completely disparate solutions
  - Each can be described as a degenerate case of the other

Internet routing scalability

- PARC CCN/NDN proposes using existing IP routing protocols to carry “content routes”
  - Define a “content” address family in BGP
  - Interest packets carry content names in URI form, ‘/’ separated
  - Forward packets based on longest-prefix match on content name
- Can we use current routing protocols to build an internet-sized NDN routing system?
  - If so, what will it take?
  - If not, what do we do instead?

BGP scaling recap

- Today's biggest Internet routing table (the DFZ) carries about $4x10^5$ routes in BGP
  - This covers about $3.8x10^9$ IPv4 addresses, $6x10^8$ hosts today
  - Prefix routing & summarization thus gives us about four orders of magnitude
- In general, the DFZ has tended to grow quadratically
  - From 50k prefixes in 1998 to 250k prefixes in 2008, to 400k today
- There are many factors that drive the growth in DFZ size up or down
  - Up: Fragmentation of prefix advertisements due to traffic engineering, multi-homing, PI addressing
  - Down: High cost of TCAM & SRAM hardware for building forwarding ASICs, cost of upgrading routing hardware globally
  - This was studied in RFC4984
- Is this the highest BGP can go?

BGP scaling recap

- Let's look strictly at BGP as a routing *protocol*
  - Ignore the requirements imposed by a single routing table
  - Ignore IP forwarding ASIC issues
- MPLS VPN has introduced significantly higher BGP scaling requirements
  - Although no single table is bigger, PEs hosting many VPNs may carry 2 million+ routes
- High-end route reflectors today can handle ~3-4 million routes
  - This seems like a good estimate of what BGP can do today
- Is it enough?

BGP scaling recap

- What prevents BGP from scaling 10x higher?
- This is both a *state* problem and a *rate* problem
- It already takes quite some time for new BGP peers to "settle" i.e. converge all their state
- Update rates are driven by a number of factors
  - Prefix count
  - Network diameter (AS path length) – not increasing in last 5 years
  - "Density" of connectivity – high in the core of self-similar networks
  - Prefix "instability" – doing better in last 5 years, MRAI?
- We can see no straightforward way to scale BGP by 10x in the short term
  - Last time, it took about 10 years to scale BGP by 10x

Size of ICN routing table?

- How big is an ICN routing table?
  - Let's look at what we need *today*
  - Assuming that future requirements growth will use up any future scalability improvements
- Some ICN schemes such as DONA use a flat namespace
  - Self-certifying, better security, more flexible
- We estimate an Internet of flat labels would have to support $O(10^{12})$ routes *today*
  - Google has indexed $>10^{12}$ URLs
  - Web measured at $5x10^{10}$ ***text pages***
- This is six orders of magnitude beyond BGP
  - … so we are not optimistic to reach that anytime soon
- We need to compress the routing table

Size of ICN routing table?

- NDN compresses the routing table using longest-prefix-match forwarding
  - Only prefixes carried in the routing table, not entire labels
- This limits the nature of content names
  - In general, data packets need to have unambiguous *names*
  - But for NDN routing, they need to start with unambiguous *prefixes*
- It is desirable to have *location-independent* data packet prefixes
  - Some literature asserts that “permanent names cannot have mutable semantics”
  - Location is a mutable semantic (both in terms of endpoint mobility as well as path changes)
  - So NDN routing cannot rely on topological prefix binding

Size of NDN routing table?

- There is already a global, unambiguous, location-independent set of names we can use as prefixes – DNS names
  - NDN efforts use DNS names as first element in prefix
- There are 13×10^7 DNS names currently assigned in the gTLDs (.com, .net, .org, .info)
  - Approx 9.5×10^7 in .com alone
  - If we add in ccTLDs, the total number is close to 2×10^8
- A NDN routing table that carried *only top-level domains as prefixes* would have to carry **N=2×10^8 routes** today
  - No useful summarization possible below this number

Size of NDN routing table?

- What if the routing table carried second-level and third-level prefixes?
  - Large providers would want to have multiple data centers that are not exact duplicates of each other
  - e.g. <disney URLs>
- Let us assume that subdomains of content would follow a *scale-free* growth pattern
  - Degree of nodes in a scale-free graph follows a power law. Probability that a node has degree d is given by

    $P(d) = c.d^{-Y}$, where $2 < Y < 3$, and c is a normalization const
  - Evidence that web sites (links between web sites) and also internet routers (and links connecting them) form scale-free graphs
  - Assume the probability that a first-level advertised domain would have *d* advertised sub-prefixes is given by this function

Size of NDN routing table

- Let number of first-level domains be N
- Expected total number of sub-prefixes advertised is given by

$$S = \sum_{i=1}^{N} i \times i^{-\gamma} = \sum_{i=1}^{N} i^{-(\gamma-1)} = H_{N,\gamma-1} \cong \zeta(\gamma - 1)$$

(last approximation holds for large values of N. Includes the top level domain)

- What is the value of the free parameter γ?
  - Internet router scale-free graph matches $\gamma = 2.1$
  - Web scale-free graph matches $\gamma = 2.4$

γ	2.1	2.2	2.3	2.4	2.5
S	10.58	5.59	3.93	3.11	2.61

  - So let's assume factor of 3, which means **N_S=6x10^8 routes**
    - 200 times greater than BGP today ☹

Other routing options?

- LISP?
  - ITR/ETRs still have to participate in a large routing database
  - But there could be something here…
- Compact routing algorithms?
  - Scale-free nature of Internet graph means compact routing has low stretch…
  - … but high message update rate, and will be driven higher by these huge prefix counts
  - No global dynamic compact routing algorithm known which can scale to these levels with reasonable stretch
- What about DHT/flat label routing algorithms?
  - Very high stretch, since the top-level ring will circle around many hundreds of times for such a large table

A path forward?

- We need more compression of the routing table.
  - At least six orders of magnitude total
  - It would be really nice if we could do it with flat content labels
- How?

- Insight 1: Topological aggregation of location-independent addresses is hard
  - There are many different statements of this property
  - Routing algorithms prefer topological aggregation
  - So let's see if we can exploit topology rather than ignore it
- Insight 2: The problem is hard because of a *combination* of things
  - A single $O(10^{12})$ database isn't impossible
  - Neither is a global routing protocol spanning 200,000 routers
  - But the combination of both of them is where the problem is
  - So let's try to decompose the problem

Our proposal, briefly

- Retain topological properties of labels used in routing
  - … and therefore compress the routing table topologically
  - *But* remove the requirement that these labels identify hosts
  - … and therefore relax the requirement that content be transferred by one-to-one transport sessions
- Don't rely on implicit hierarchy (longest-prefix-match)
  - Partially-ordered list of labels
  - Routing selects the one with highest precedence for which there is a route
- Use a translation lookup to convert from content name to routing label(s)
  - Translate from "what this is" to "where it can be found"
- Use partially ordered list of *content labels* to compress the content-name→routing-label lookup database
  - Once again, use exact match instead of longest-prefix match
- More details in an upcoming paper

QUESTIONS?

BACKUP SLIDES

Routing Labels

- Interest packets will carry a partially ordered set of *routing labels*
  - Routing labels identify *where* the data in question is present
  - Label could represent a server, a subnet, a data center, a large aggregate route
- Routers forward interest packets by matching routing labels to routes, which are carried by a routing protocol
  - Preferentially forward towards the route matching highest-precedence label
  - But sophisticated forwarding may do better, spreading interests across:
    - Equal-cost next-hops to the same route
    - Multiple routes to labels of equal precedence
    - Routes of different cost, unequally/proportionally
    - Policy-based forwarding
- Why multiple routing labels?
  - As an explicit aggregation mechanism, instead of longest-match prefix
  - Routes for more specific labels may be advertised only closer to the destination, with more aggregate routes being advertised more broadly
  - A single IP address label with prefix routing can be used, but is less flexible
  - Plus this makes the label lookup service easier to manage, see later

Routing Label Examples

- Flat labels, can be assigned by provider
  - But must be globally unique

- Single ACME CDN server in VA, USA:
  - `acme.cdn_L_ACDN_US_001C44_1A556C`
- CDN array in VA, USA (behind load-balancer):
  - `acme.cdn_L_ACDN_US_VA3_114`
- Gateway to entire ACME VA data center 3
  - `acme.cdn_L_ACDN_US_VA3`
- Gateway to ACME CDN – Eastern US AS
  - `acme.cdn_L_ACDN_US_NAE`

Routing Label Lookup

- Endpoint looks up the routing labels associated with a content name, and puts the labels into the Interest msg
  - Lookup could be done at gateway or intermediate hops like LISP
  - Endpoints or intermediaries can cache the locator labels and apply to similar content requests
- Interest packets carry *both* content ID/name *and* routing labels
  - Object lookup and caching done on basis of Content ID
  - Only routing/forwarding on basis of routing labels
  - But gateways/load-balancers can use Content-ID as well
- Lookup service maps content ID to routing labels
  - Mapping database distributed globally
  - Labels returned are not specific to sender
    - Which is why this is *not* source routing

Content-ID

- Content is identified by a Content-ID
- This is a partially-ordered set of *content label* strings
  - Highest precedence label uniquely identifies the object
  - Lower precedence labels describe *content associations* of the object
    - e.g. groups of content which contain this object
    - e.g. content labels that would be co-located with or close to this object
- Why multiple labels?
  - Because we want to scale the content lookup database in the same way as the routing table, but still want to use exact match
  - A single content-ID string with longest prefix match can be used, but is less flexible
- Content-ID is defined by the content publisher
  - Public identifier of the content object
  - Publisher should select labels such that the lifetime of the Content-ID is at least as long as the lifetime of the content object

Content-ID Examples

- Content-ID supports any of these formats
- Purely hierarchical strings (like NDN)
  - `G1 := {ashokn.org/blog/2011/Sep/19/01 | ashokn.org/blog | ashokn.org}`
- Non-hierarchical strings
  - `G2 := {cnn.com/video/1109/01a33v.mp4 | cnn.com/media/video, cnn.com/ASHOK | cnn.com}`
- Strings + flat IDs (like hashes)
  - `G3 := {CNN_1A33510DA92155E1 | CNN_0D3321A9E1DA3870 | CNN_VIDEO }`
- Purely flat IDs (like DONA)
  - `G4 := {14C61A00_080327713ADE237F | 14C61A00_298B3CFF2116AEDD }`

Label Lookup Database

- The ID→Label Mapping database maps *individual content labels* to one or more routing labels
- Content-ID is *locatable* if the intersection of content labels and lookup database is non-empty
  - Only aggregate content labels need to be present in the database
  - This allows a lookup database of $O(10^8)$ entries to support $O(10^{12})$ content objects with flat label matching (shown in the paper)
- Lookup database is replicated globally
  - Entire database replicated at Tier-1 sites or DHT
  - Lower-tier sites can implement caches
- Lookup database populated by content publishers
  - Publishers who use CDN hosting service get CDN routing labels from the hosting providers (could be multiple for large hosts)
  - Self-publishers use their own routing labels
  - Can combine the two for both efficiency and generality

Example 1

- Example 1: Ashok has a vanity website `ashokn.org`
- Buys hosting from GoMommy.com
- GoMommy gives Ashok content labels to use for his content
- Ashok associates his content with GoMommy labels
  - `ID := {ashokn.org/blog/2011/Nov21/main.html | gomommy.com/cust/useast/edc1/dcsc14 | gomommy.com/cust/useast | ashokn.org}`
- GoMommy has inserted routing labels into the lookup database:
  - `gomommy.com/cust/east → { gomommy.com_L_USE1_DC1 | gomommy.com_L_US }`
- Ashok doesn't insert anything into the lookup database…
- … BUT, his Content-IDs are bound to GoMommy
  - And if he changes his hosting, the Content-IDs need to be changed

Example 1

- Client gets ID1 from an email
- Lookup returns labels {gomommy.com_L_USE1_DC1 |gomommy.com_L_US}
  - No routing labels for first, second or fourth content labels
  - Interest packet carries the ID and routing labels
- GoMommy has a global routing advertisement for their US AS gomommy.com_L_US
  - All routers route towards their US AS using the less specific routing label
- Within their AS, GoMommy has a route advertisement for gomommy.com_L_USE1_DC1 towards their US East Data Center 1
  - Routers within the AS can route towards the more specific routing label
- The load balancer which enters their data center sees the Content-ID
  - … and determines that the content is on the “Server 14” cluster, routing the interest towards the load balancer in front of it
- The load balancer managing the Server 14 cluster can look at the content label for ashokn.org and route towards the specific server that has that content

Example 2

- Example 2: Jeff runs a small business Goldberg.com
- Buys hosting from CloudHost Inc
  - CloudHost is an international hosting company which has four datacenters spread across the world
  - They advertise the same routing label `cloudhost.inc_L_glbl` from different hosting ASes into the global Internet
- CloudHost gives Jeff routing labels for the server(s) hosting his content
- Jeff adds an entry into the lookup database for his company
  - `goldberg.com → { cloudhost.inc_L_clust44 | cloudhost.inc_L_glbl }`
- Jeff's Content-ID labels now reference his own entry
  - `ID := {goldberg.com/sales/contact.htm | goldberg.com}`
- This is a permanent Content-ID
  - If Jeff changes his hosting, he simply updates his entry in the lookup database
- Routing follows the same flow as in Example 1
  - In the global network, interests are automatically routed towards the closest CloudHost AS advertising `cloudhost.inc_L_glbl`
  - Within CloudHost's AS, the more specific label routes the interest towards server cluster44.
  - Server cluster44 can use the Content-ID to determine the content to be served

Example 2b

- Example 2b: Jeff adds Level-4 as a second hosting provider
- They give Jeff new routing labels
- Jeff updates his entry in the lookup database
  - He's sure to keep the globally routed labels from both providers at the same precedence
  - `goldberg.com → { cloudhost.inc_L_clust1, level4.cdn_srvr1 | level4.cdn_srvclust | cloudhost.inc_L_glbl, level4.cdn }`
- Distant clients will automatically select the closest provider with Jeff's content – CloudHost or Level-4
- Once the Interest enters the provider's AS, more specific routing labels or content labels route the interest to the content
  - And the routing labels from the other provider are automatically ignored