Scalability of Routing: Compactness and Dynamics

Dmitri Krioukov (CAIDA)

dima@caida.org

Kevin Fall (Intel Research) and kc claffy (CAIDA)

IETF-67

Shocking news

There exist routing algorithms such that even if all of 2^{128} IPv6 'nodes' are completely de-aggregated (i.e., all IPv6 addresses are used as flat IDs), the 'DFZ' routing tables still contain less than $128^2 \sim 16{,}000$ entries (~1000 entries for IPv4)

Caveats of the shocking news

- Assumptions about Internet topology [it's scale-free] (realistic)
- Assumptions about possibilities to not always follow shortest paths [stretch >1] (realistic)
- Assumptions about having the 'full view' of the graph [the network is static] (unrealistic)

Outline

- What causes scalability problems (in a nutshell)
- Why network topology is important
- Why static routing scales almost infinitely on realistic topologies
- Why name-dependent compact routing is a generalization of ‘hierarchical’ routing
- Why and how name-independent compact routing delivers the desired locator/ID split
- Why locator/ID split (name-independence) can NOT buy us a free lunch
- How you can help
- Why you can cheer up

Major causes of scalability problems

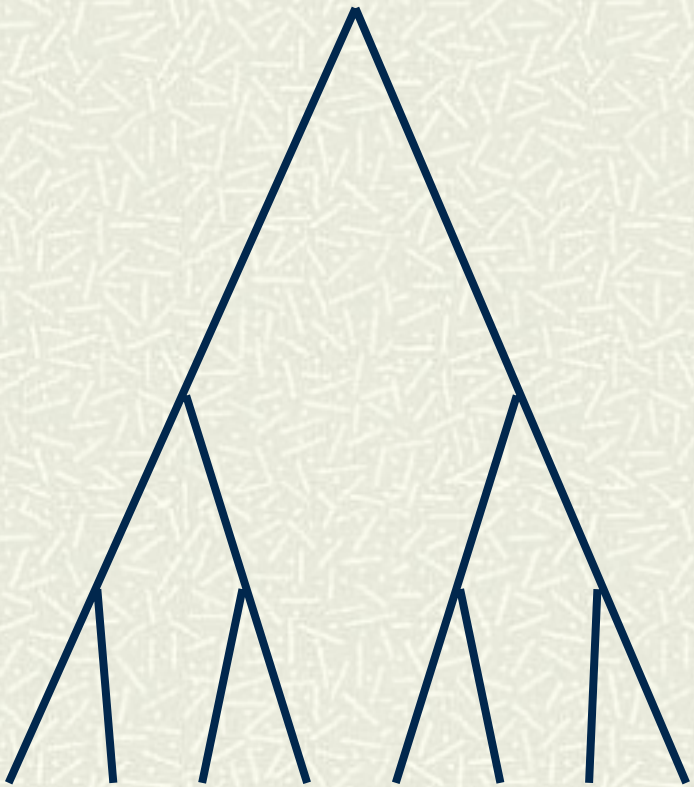
- The Internet is:
  - large and growing
  - dynamic and more dynamic
- Address de-aggregation
  - lots of reasons
  - most of them are well-documented

Extreme form of de-aggregation

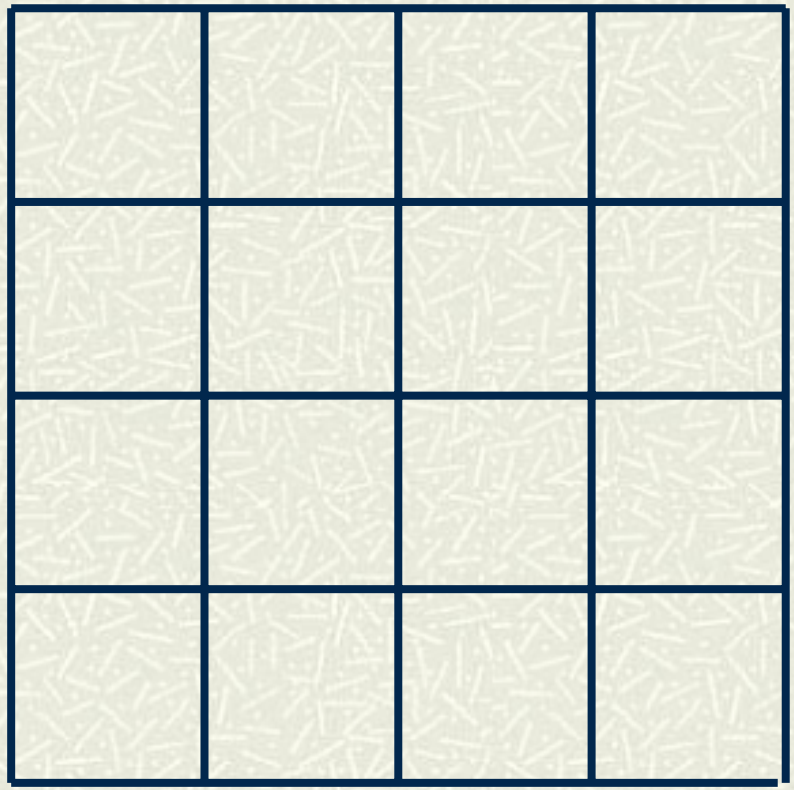
- All IPv[46] addresses need to be represented as individual nodes of the global Internet topology graph

Extreme forms of aggregation

- Trees



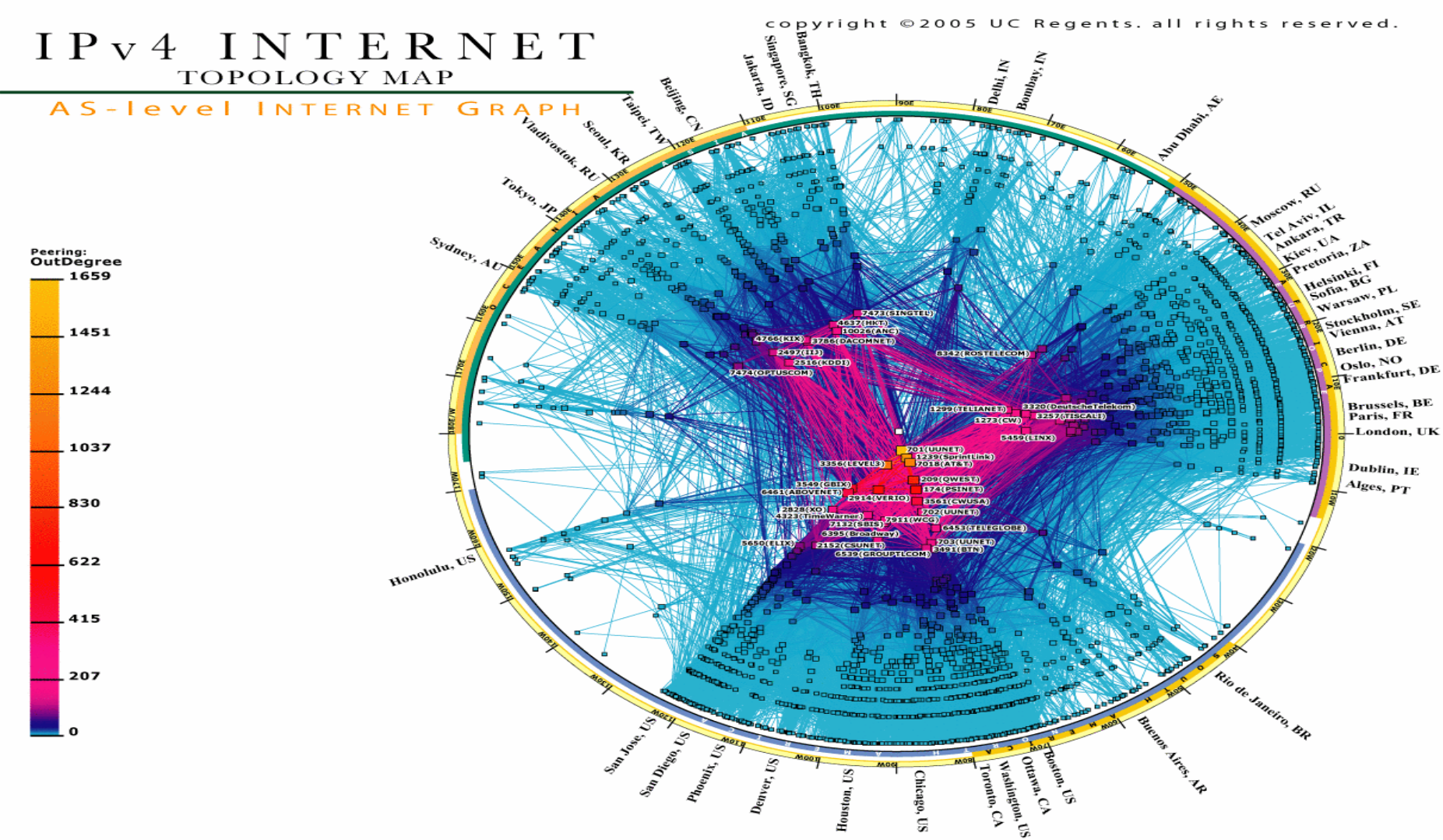
- Grids



Shocking news (of 1999)

- The Internet is neither a tree or a grid
- The Internet is…

What we've got



Scale-free topologies

- Dangerous waters
  - lots of polemics about if the Internet is ‘scale-free’ or not and about what ‘scale-free’ means
  - lots of recent work on Internet topology data analysis
- But all parties (and data ☺) seem to agree that the Internet, both at the router- and AS-levels, has:
  - heavy-tailed degree distributions (power-laws)
    - small average shortest path lengths, as a consequence
  - strong clustering

Properties of scale-free networks

- They do not allow for efficient address aggregation
- But they do allow for extremely efficient static compact routing

Compact routing

- Construct routing algorithms such that
  - given the full view of the network topology
  - the trade-off between routing table sizes and stretch is balanced in the most efficient way
- Stretch is a measure of the increase of lengths of paths produced by a routing algorithm compared to shortest path lengths
  - compact routing algorithms make routing table sizes compact by means of omitting some details of the network topology in an efficient way such that the resulting path length increase stays small
- A routing algorithm is compact if (definition):
  - Node address and packet header sizes scale polylogarithmically
  - Routing table sizes scale sublinearly
  - Stretch is a constant (does not grow with the network size at all!)

Two classes of compact routing

- Universal
  - applicable to ALL graph
- Specialized
  - utilize peculiarities of network topologies of a certain type in order to achieve better performance

Examples of why topology matters

- Routing on the nicest graphs (trees or grids)
  - logarithmic address sizes
  - logarithmic routing table sizes
  - shortest path routing
- Routing on all graphs
  - shortest path routing
    - cannot route with sublinear routing table sizes
  - need to allow for at least 3-time path length increase
    - to route with sublinear ($\sqrt{n}$) routing table sizes

TZ scheme

- The scheme is the first optimal (upper bounds = lower bounds) universal compact routing scheme
  - stretch is 3
  - routing table size is $O(\sqrt{n})$

TZ scheme internals

- **neighborhoods (clusters)**: my neighborhood is a set of nodes closer to me than to their closest landmarks
- **landmark set (LS) construction**: iterations of random selections of nodes to guarantee the right balance between the neighborhood size ($O(\sqrt{n})$) and LS size ($O(\sqrt{n})$)
- **routing table**: shortest paths to the nodes in the neighborhood and landmarks
- **naming**: original node ID, its closest landmark ID, the ID of the closest landmark's port lying on the shortest path from the landmark to the node
- **forwarding at node *v* to destination *d*:**
  - if $v = d$, done
  - if d is in the routing table (neighbor or landmark), use it to route along the shortest path
  - if v is d's landmark, the outgoing port is in the destination address in the packet, use it to route along the shortest path
  - default: d's landmark in the destination address in the packet and the route to this landmark is in the routing table, use it

TZ scheme and scale-free graphs

- We found that the TZ scheme performs essentially optimally on scale-free (Internet-like) graphs: all other graphs yield worse results
- This finding was the first indicator that scale-free topologies are particularly ‘well-structured’ and allow for outstanding routing performance

BC scheme

- The scheme is the first scheme specialized for scale-free graphs
- Internals
  - find the highest-degree node in the graph
  - grow the shortest path tree rooted at it
  - find the core, which is all nodes located within maximum distance d from the highest degree node
  - grow small number (e) of trees to cover all the edges that do not belong to the main tree and lie outside of the core
    - the larger d, the smaller e
  - use known ultra-compact routing algorithms to route on these trees
- Guarantees
  - Additive stretch d
  - Routing table sizes $O(e \log^2 n)$

Why BC scheme is infinitely scalable

- Diameter of scale-free graphs grows as $O(\log n)$
- If we allow for a logarithmic additive stretch d, we can let the core grow to encompass the whole graph in order to make e constant
- Routing table size is thus $O(\log^2 n)$
- And $\log^2 2^{128} = 128^2$
- Fed with the current Internet AS-level topology, the BC scheme produces
  - routine table with 22 entries (1025 bits)
  - multiplicative stretch of 1.01

A bit of pessimism

- The algorithms are essentially static
  - Topology pre-processing (pre-computation) costs are not considered
  - Distributed implementations are possible, but the bounds for the number of messages they need to generate in order to process a topology change are not considered

Another classification of compact routing algorithms

- Name-dependent
  - routing algorithms require special forms of node addressing in order to embed useful topological information in addresses
  - in other (Yakov's) words: "addressing follows topology"
  - if topology admits aggregation, we have a generalization of 'hierarchical' routing
- Name-independent
  - routing algorithms can work on arbitrary topologies with arbitrary node addresses/IDs
  - in other (contrary to Yakov's) words: neither addressing follows topology, nor topology follows addressing
  - name-independent compact routing can thus route of flat IDs

Shocking news on name-independence

There exist universal name-independent routing algorithms with the same performance guarantees as their name-dependent counterparts

Name-independence vs. locator/ID split

- Any name-independent compact routing algorithm implements a form of locator/ID split by having
  - name-dependent compact routing using locators underneath, and on top of that:
  - ID-to-locator-mapping information, efficiently distributed among nodes so that not to break performance guarantees

Why people want to split locators and IDs

- If addresses follow topology, then as soon as topology changes addresses must change as well, which does not scale
- For better scaling, we thus need to split the location and ID parts in our addressing architecture. Period.
- BUT THERE IS NO FREE LUNCH: if the network is dynamic (links come and go, nodes move), we still need to have up-to-date information on where the destination ID currently is
- *In addition* to properly updating locators (to follow topology), we also need to dynamically update the ID-to-locator mapping (distributed) database
- These considerations directly apply to name-independent compact routing

Abraham *et al.* scheme

- The scheme is the first optimal (upper bounds = lower bounds) universal *name-independent* compact routing scheme
  - stretch is 3
  - routing table size is $O(\sqrt{n})$

Abraham internals for metric spaces

- **neighborhoods (balls)**: my neighborhood is a set of $O(\sqrt{n})$ nodes closest to me
- **coloring**: color every node by one of $O(\sqrt{n})$ colors ($O(\sqrt{n})$ color-sets containing $O(\sqrt{n})$ nodes each), s.t. every node's neighborhood contains at least one representative of every color (all colors are 'everywhere dense' in the metric space)
- **hashing names to colors**: just use first $\log(\sqrt{n})$ bits of some hash function values (it's ok w.h.p.)
- **routing table**: nodes in the neighborhood and nodes of the same color
- **forwarding at node *v* to destination *d*:**
  - if $v = d$, done
  - if d is in the routing table (neighbor or v's color), use it to route along the shortest path
  - default: forward to v's closest neighbor of d's color
- has been implemented and deployed (overlay 'Tulip' on PlanetLab)

Abraham internals for graphs

- **LS set**: all nodes *l* of one selected color
- **ultra-compact routing on trees**: every node resides in $O(\sqrt{n})$ of such trees *T*:
- **routing table of *v***:
  - shortest-path links to neighbors
  - *T(l,v)* for all landmarks *l* (i.e., the routing table produced for *v* by tree-routing on the shortest-path tree rooted at *l*)
  - *T(x,v)* for all neighbors *x*
  - for all nodes *u* of *v*'s color, either (whatever corresponds to a shorter path):
    - info for path in $T(l_u)$ ($v \rightarrow T(l_u) \rightarrow u$), or
    - info for path via *w*, where *w* is s.t.: 1) *v* is a *w*'s neighbor, and 2) *w*'s and *u*'s neighborhoods are one hop away from each other ($v \rightarrow w \rightarrow x \rightarrow y \rightarrow u$, where *v*,*x* are *w*'s neighbors and *y* is *u*'s neighbor)
- **forwarding at node *v* to destination *d*:**
  - if $v = d$, done
  - if *d* is in the routing table (neighbor or landmark or *v*'s color), use it
  - default: forward to *v*'s closest neighbor of *d*'s color

Abraham on the Internet topology

- Routing table sizes are still small
  - ~6k entries and ~300k bits
- Stretch is somewhat less exciting
  - ~1.5
- No estimates of (pre-)computation or adaptation costs (future work)

Conclusions

- Main non-problems with routing scalability
  - **Routing table sizes**: routing tables can be made extremely succinct by using modern compact routing algorithms inducing only infinitesimal path length increase on realistic topologies
- Main problems with routing scalability
  - **Full view**: all known routing algorithms, either envisioned in theory or used in practice today, require that all nodes (in a routing 'domain') have the full view of the network topology graph (link-state, compact routing) or at least of the distances to all the destinations (distance- or path-vector)
  - **Dynamics**: this topology information must be promptly updated, at all nodes, if the network is dynamic

How you (engineering and operation communities) can help

- Bring more awareness (e.g., by publishing RFCs, papers, etc.) about:
  - that the problem exists, and
  - its specific details (e.g., how large FIBs will be or how much churn BGP will produce in X years, etc.)

How you can help

- Build bridges to other communities: the research community in the first place
  - the problem is too hard
    - if the realistic topologies delivered the worst case of the known lower bounds for routing convergence/adaptation costs, then the problem would be fundamentally unsolvable
  - the complete knowledge required to solve the problem is distributed across different communities and across different groups within the communities
  - the problem can hardly be solved by one community or group in isolation
  - research community is largely unaware of the problem and its details

How you can help

- Talk to your favorite funding agency
  - as a concerted inter-community research effort is needed
- Do research!

A bit of optimism

- The core of the problem appears to be that we need to have the full up-to-date view of a dynamic network
- In 1966, Stanley Milgram performed experiments with packet forwarding across social (acquaintance) networks
- The experiments demonstrated that humans do not need the global view to route packets efficiently over dynamic topologies that have the macroscopic structure similar to the Internet's
- Can humans build routing devices that would do the same is the question