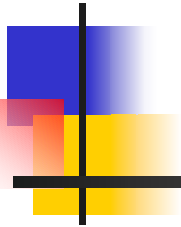


# Components of Routing Table Growth



Harsha Narayan  
(U. of California, San Diego)

Joint work with Ramesh Govindan (U. of Southern  
California) and George Varghese (U. of California, San  
Diego)



# The Goal

---

- To understand how **the routing table is growing**
- We will **develop a model** of the routing table
- The model will help us understand
  - how and why the routing table is growing

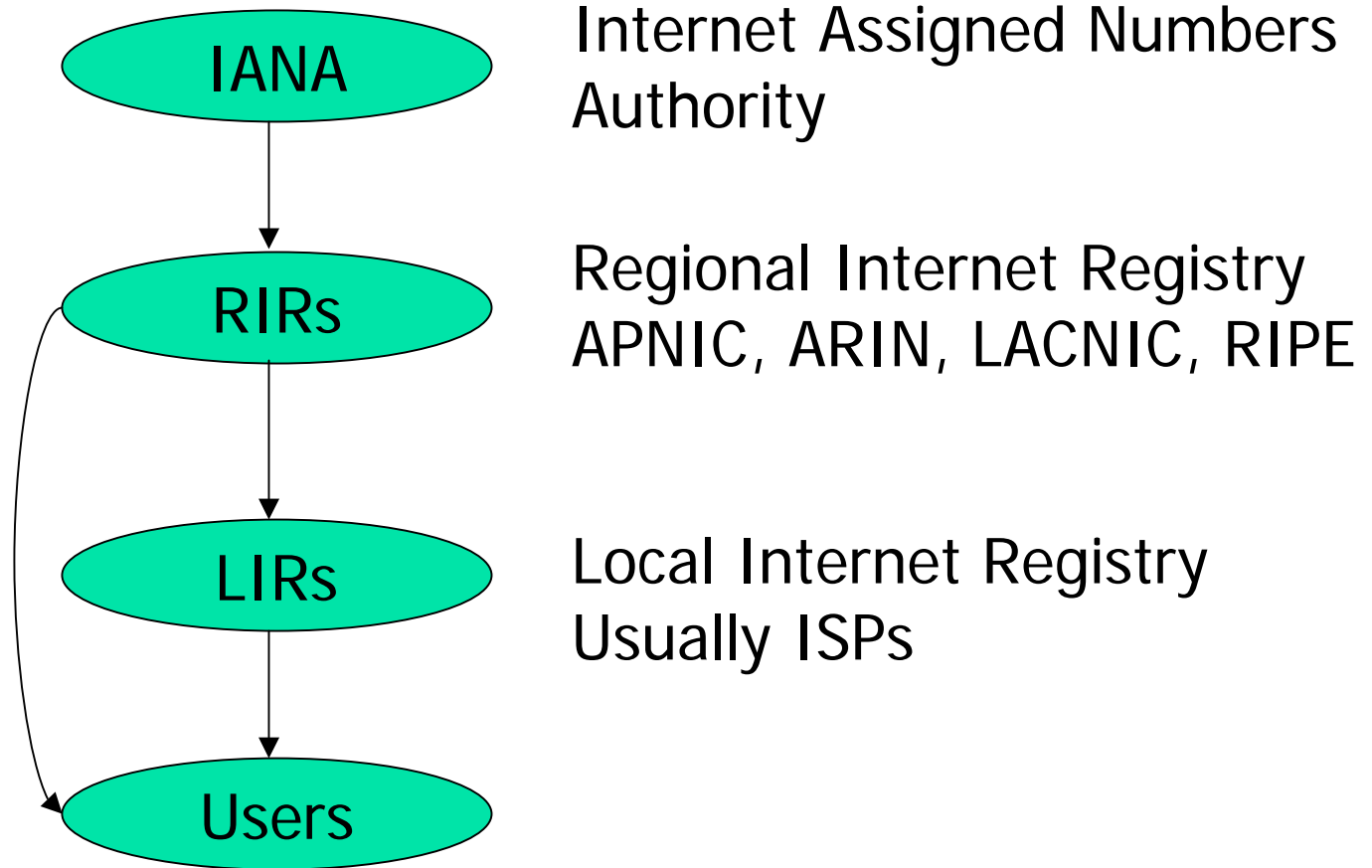


# Outline of the talk

---

- Part 1: Address Allocation and Routing Practice
- Part 2: The Model: ARAM
- Part 3: Quantifying Current Practices

# Address Management Hierarchy

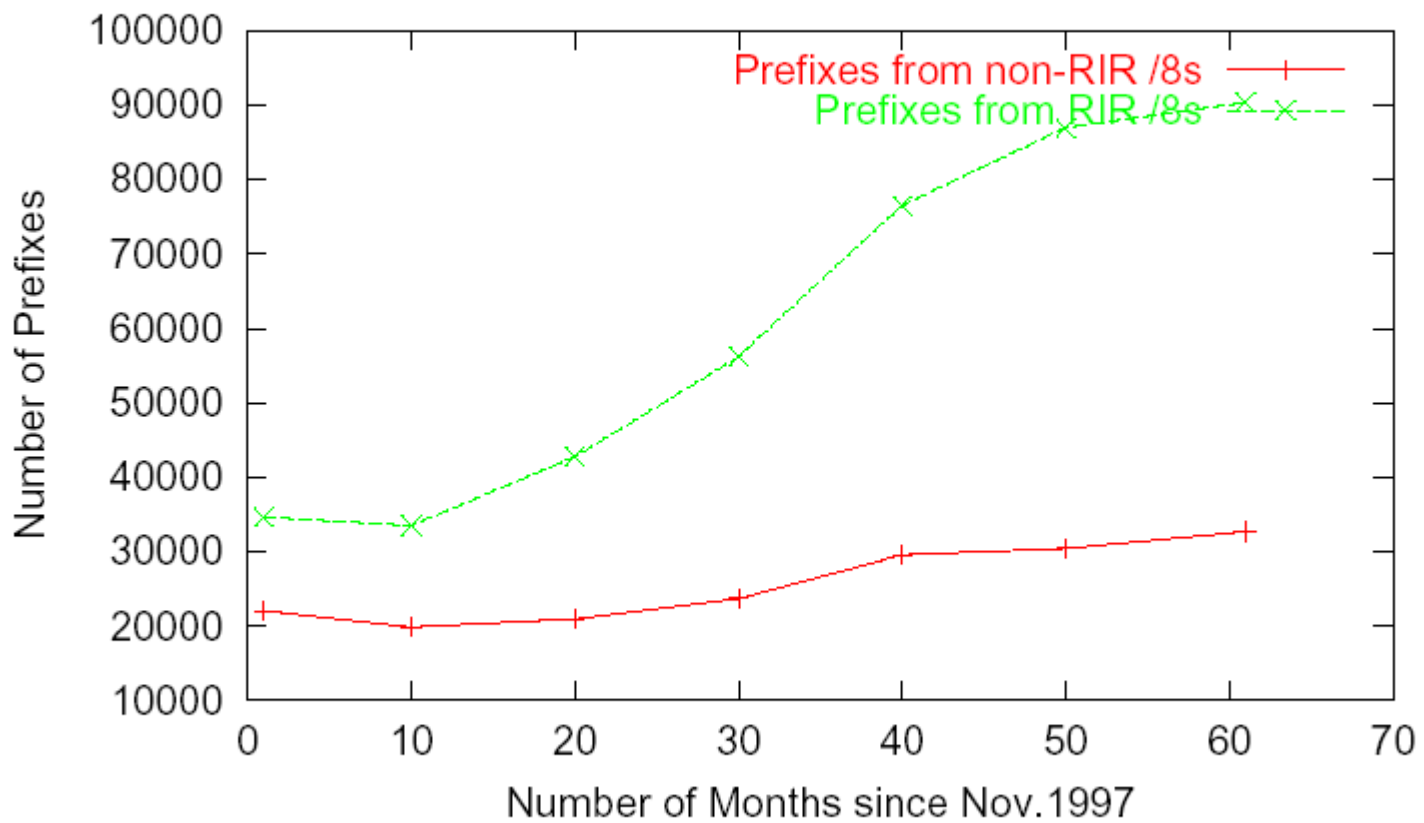




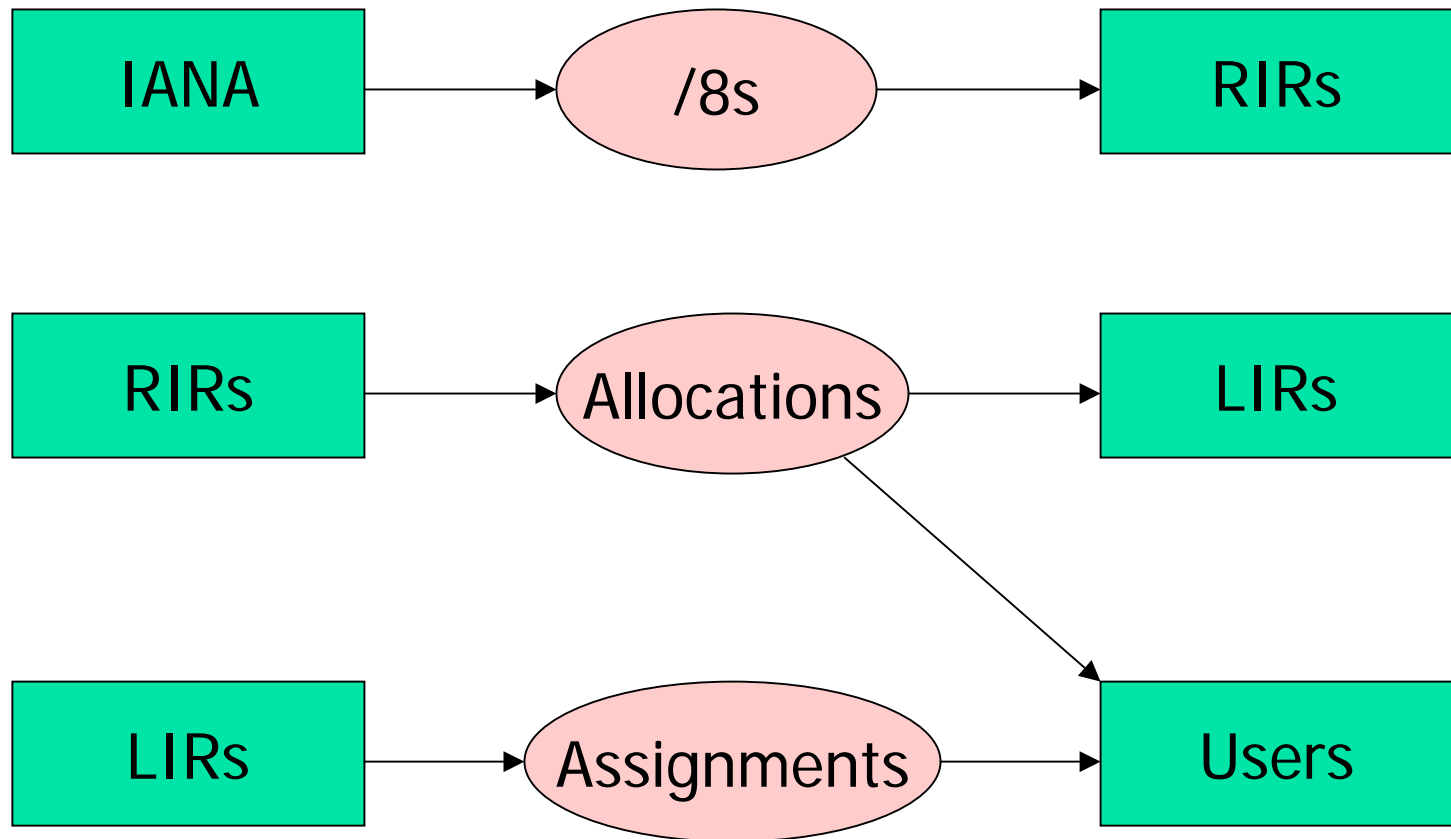
## Note

---

- We ignore that part of the routing table which developed before the establishment of the RIRs.
- This part has almost stopped growing.



# Address Allocation Hierarchy





# Depth of a prefix

---

1.0.0.0/10      Depth 0

1.0.0.0/12      Depth 1

1.0.0.0/17      Depth 2

1.0.0.0/24    and 1 more      Depth 3

Depth of a routing table prefix is the **number of less specific prefixes** in the routing table.





# Mainly Two Depths

---

- We model only two depths
  - Over the last 5 years, at least 90% of the routing table is found at Depth 0 and Depth 1 (data in next slide)



Month 1 = November 1997.

---

Month	Percentage of Routing table at Depth > 1
1	9%
10	8%
20	10%
30	10%
40	9%
50	10%
61	9%

← Always less than  
Or equal to 10%

# Important Example: Splitting and Spawning

## Allocations

1.4.0.0/14 (Intact) →

2.8.0.0/13 (Split) →

Spawned prefixes

## Routing Table

1.4.0.0/14

2.8.0.0/14  
2.12.0.0/15

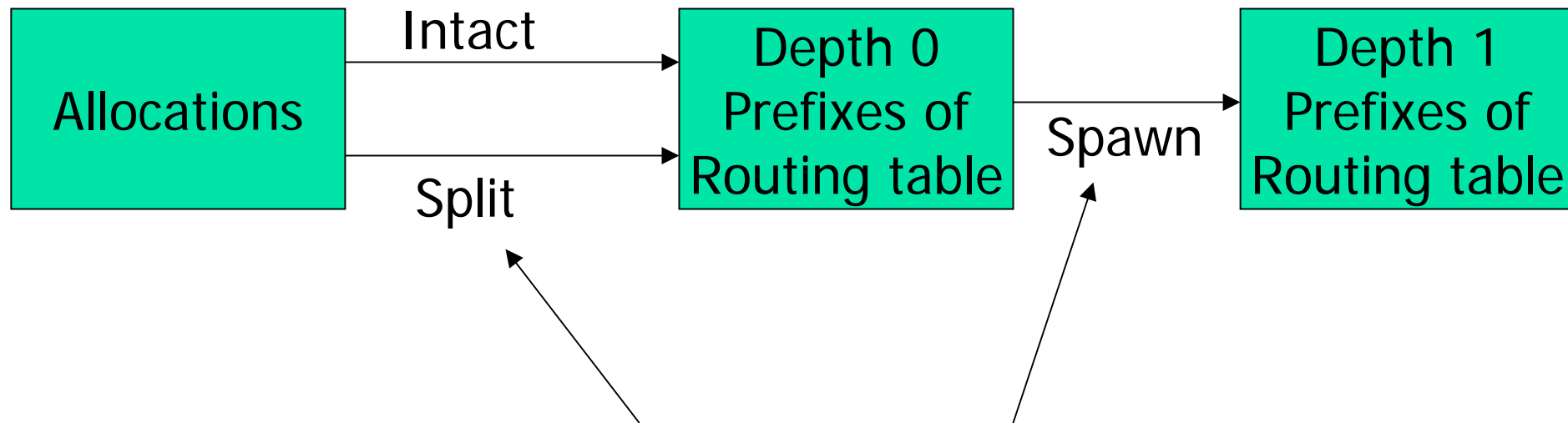
1.5.0.0/16

2.12.18.0/24

Depth 0

Depth 1

# Routing Practice



Due to Load sharing, multihoming assignments etc.



# Outline of the talk

---

- Part 1: Address Allocation and Routing Practice
- **Part 2: The Model: ARAM**
- Part 3: Quantifying Current Practices



# Our Routing Table Model (ARAM)

---

- **Input:** The number of allocations.
- **Output:** A routing table which is **structurally similar** to the real routing table
  - By structurally similar, we mean that we preserve **prefix relationships**, though prefixes need not be exactly the same.



# ARAM Overview

---

Allocation Practice → Allocations from /8s (Stage 1)

Routing Practice → Splitting (Stage 2)  
Routing Practice → Spawning (Stage 3)

Stages 1 and 2 create depth zero of the routing table.

Stage 3 creates depth one of the routing table.

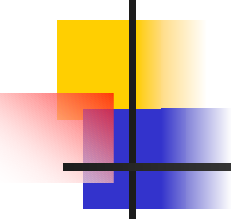


# ARAM Stage 1: Modeling Allocation Practice

---

- Given the number of allocations, our model generates them. How?
- Using curve-fitting software, we found that
  - Allocation prefix length distribution (number of allocations per allocation size) can be approximated by a power law:  $Y = (X-9)^{3.4}$
  - **X=prefix length of allocation**
  - **Y=number of allocations**
  - Basic intuition: more allocations associated with longer prefixes eg. Number of /12s=42  
Number of /15s=442.





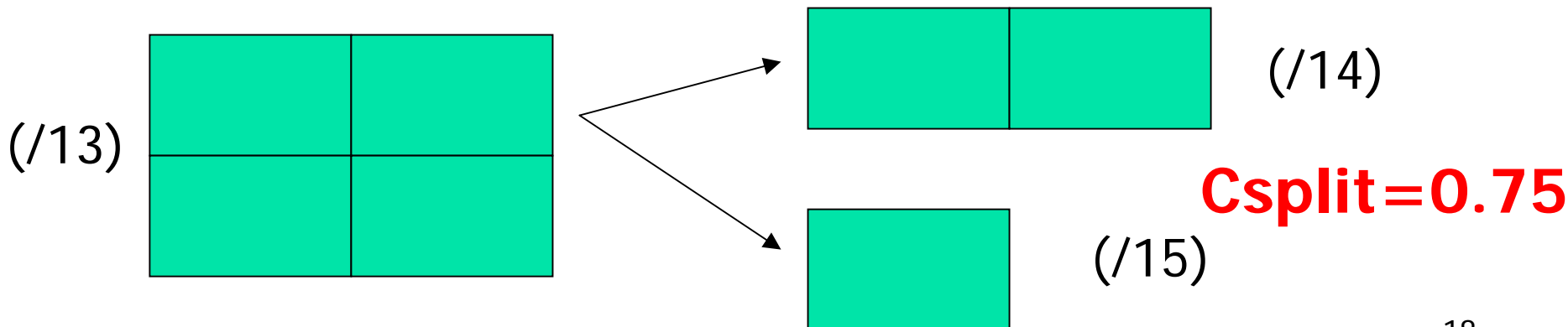
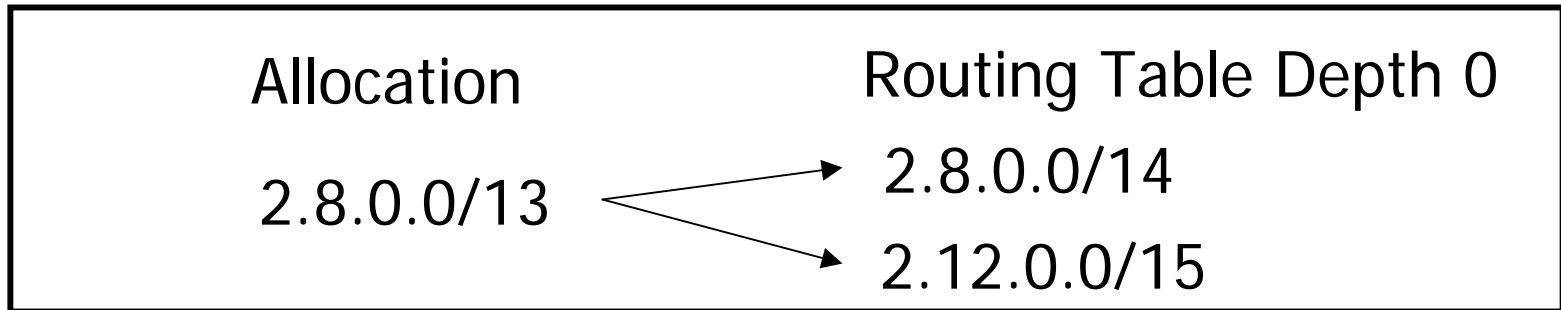
# ARAM Stage 2: Modeling Routing Practice: Splitting

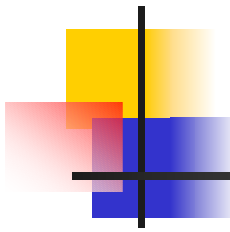
---

- We observed that not all ISPs/LIRs announce their allocations **intact**
  - Some allocations are **split** into several prefixes. Only a fraction **Fsplit** (about 20% today) of allocations split.
- Other observation:
  - A splitting allocation is not covered entirely by the split prefixes – all the available addresses need not be routed. **Csplit** – fraction of advertised address space, is around 70%.

# Example of the parameter - Csplit

The "C" is for "coverage" – fraction of advertised address space





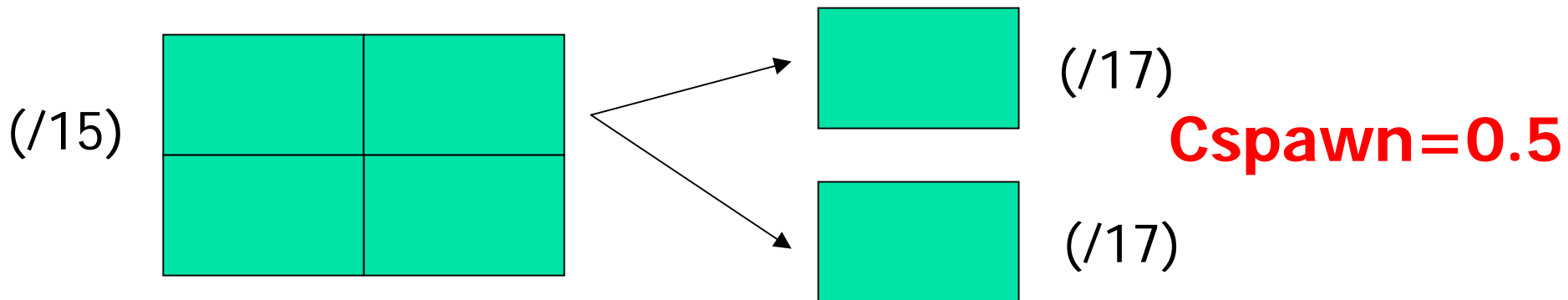
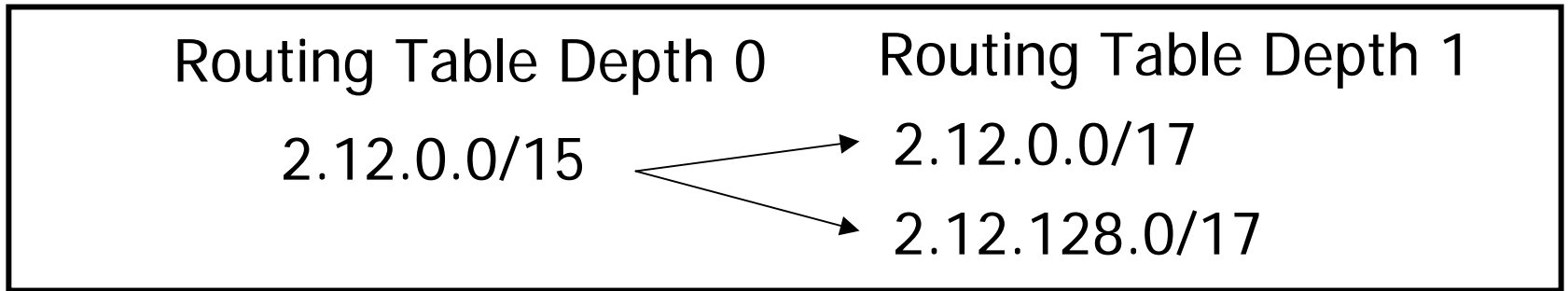
# ARAM Stage 3: Modeling Routing Practice: Spawning

---

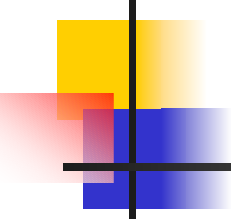
- Some **depth zero prefixes spawn the depth one prefixes** of the routing table.
- Not all depth zero prefixes spawn
  - For example, direct allocations to end-users need not spawn unless they want to do load sharing
  - Some ISPs do not make multihoming assignments
- Just like splitting, there are two parameters:
  - **Fspawn**: fraction of intact or split allocations (I.e. a Depth 0 prefix) which spawn a prefix
  - **Cspawn**: fraction of spawned address space.

# Example of the parameter - Cspawn

The "C" is for "coverage" – fraction of advertised address space



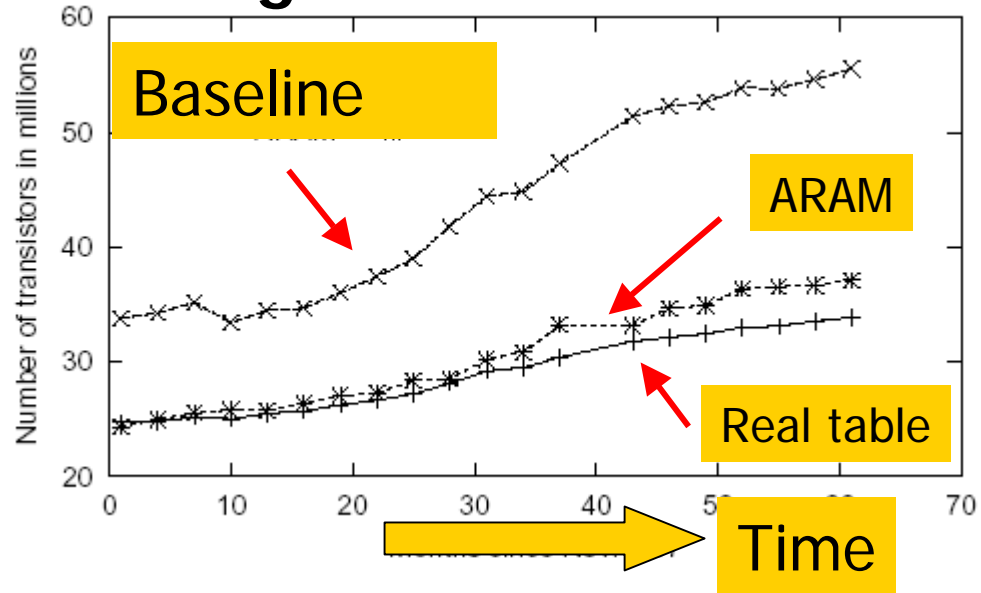
# Validation of the model

- 
- We took 20 snapshots of the real routing table at 3-month intervals.
  - For each snapshot we found the number of allocations used to generate the snapshot.
  - We used this number as input to ARAM to produce an output routing table.
  - We compared ARAM's table to the real routing table using 5 measures. To help the comparison we used a baseline model.
  - The baseline model generates prefixes at random given the prefix length distribution of the routing table as input.

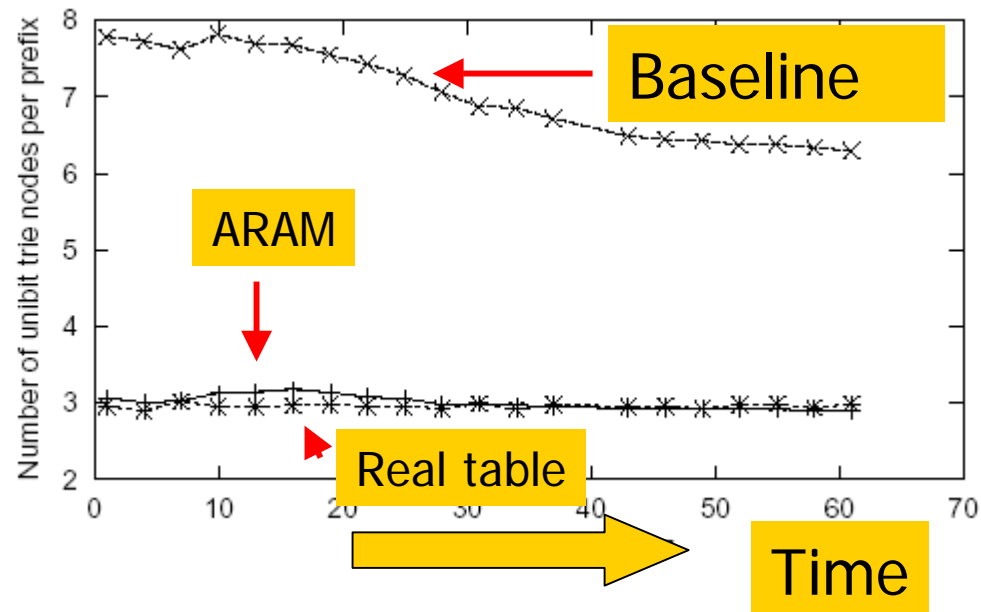
# ARAM does well

- The 5 measures were:
  1. Total number of prefixes (routing table size).
  2. Depth ratio (Depth 0 / Depth 1).
  3. Prefix length distribution
  4. Sparseness: **Number of Unibit trie nodes per prefix**. The routing table can be drawn as a binary tree - called a unibit trie.
  5. Size of data structure used in the forwarding table in routers.

## Forwarding table data structure

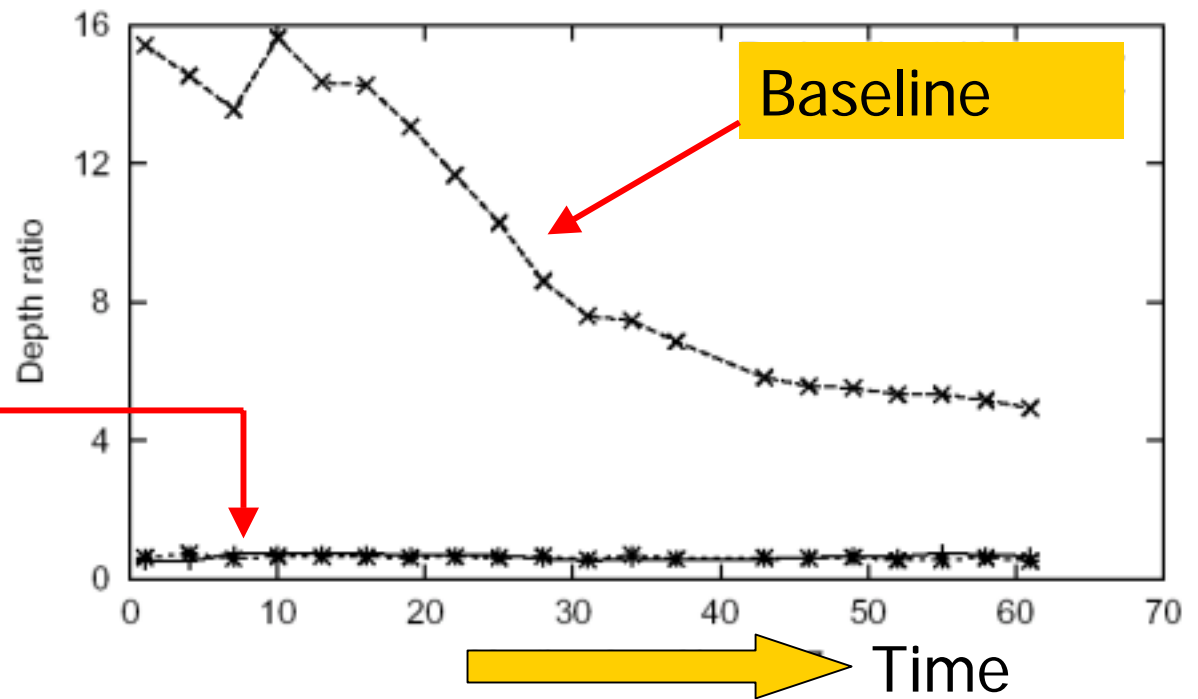


## Sparseness of Routing tree

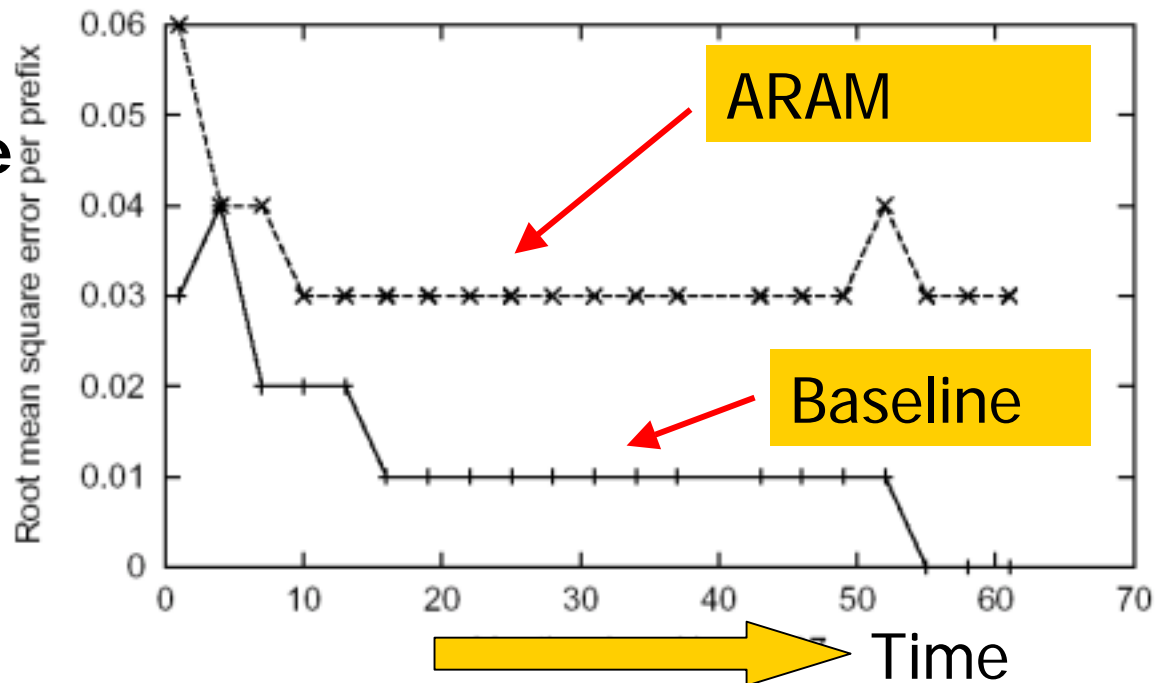


## Depth ratio

ARAM and Real table – very close. The two lines look like one.



## Root mean square error in Prefix length distribution





# Outline of the talk

---

- Part 1: Address Allocation and Routing Practice
  - Part 2: The Model: ARAM
  - Part 3: Quantifying Current Practices
- 
- Now that we have described the model
    - We can use the model's language to describe and quantify allocation and routing practice





## We will see:

---

- Why splitting and spawning happen?
- Quantify splitting and spawning.
- Extent of Load Sharing.
- Can RIRs help reduce routing table size?



# Why does splitting happen?

---

- **ISP-level Load Sharing/Load Balancing.**
  - An ISP shares traffic across multiple upstreams.
  - Or a large ISP may have some kind of internal partitioning.
  - There are also Cable ISPs which advertise large number of split prefixes.
- End-users with direct allocations from RIRs
  - use multiple ISPs.
  - use diverse routing with a large ISP
- Though it would be desirable to quantify the prevalence of the above practices, we cannot do this because there are no particular signatures.



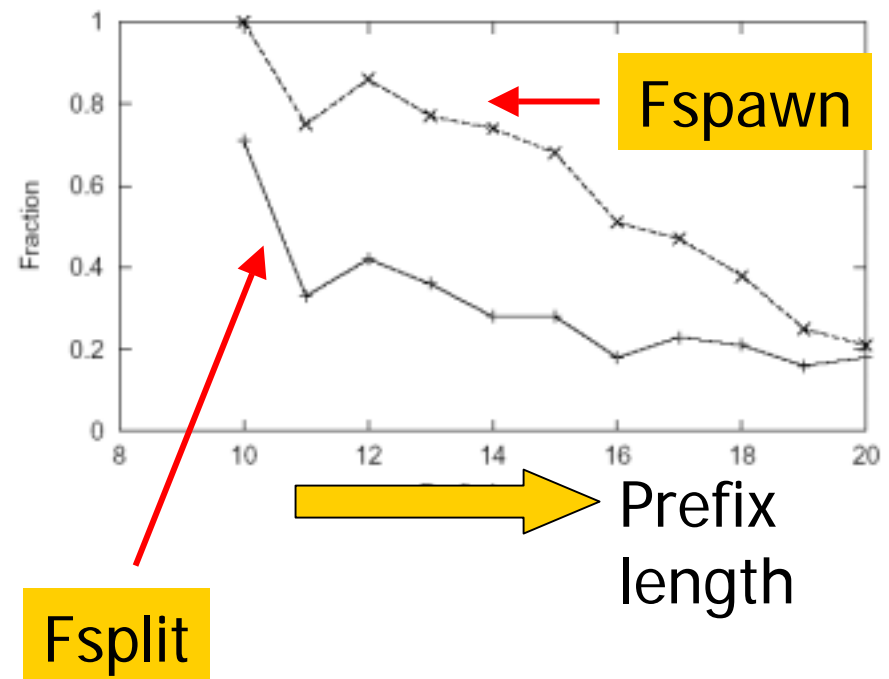
# Why does spawning happen?

---

- ISPs do load sharing of intact allocations.
- ISPs make multihoming assignments to customers. These assignments are then advertised through different providers for backup connectivity or load sharing.

# An observation about Fsplit and Fspawn

- Larger allocations are more likely to split.
- Shorter Depth 0 prefixes are more likely to spawn.
- **As the prefix length increases Fsplit and Fspawn decrease.**

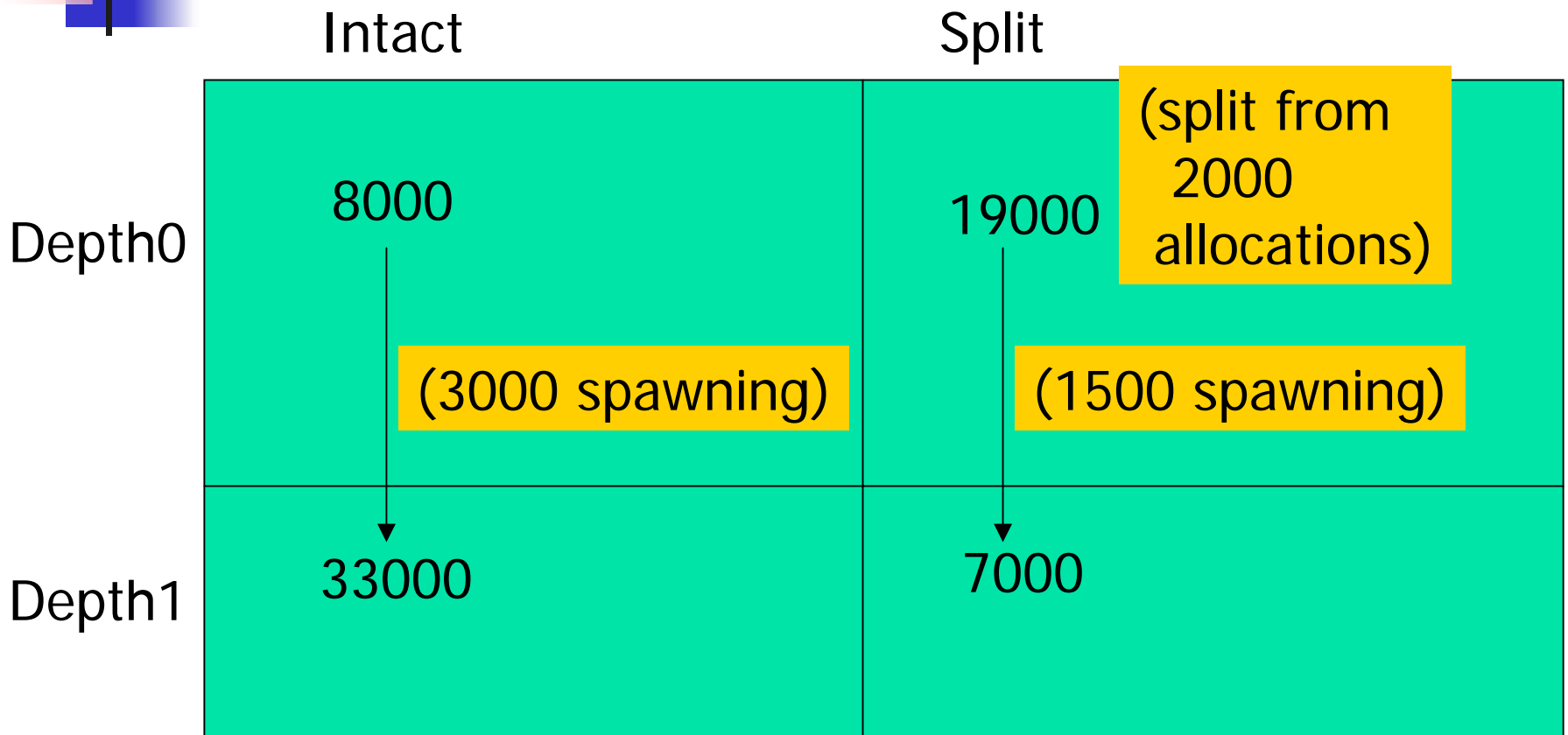


# Another View of Splitting and Spawning

	Intact	Split
Depth 0	11%	26%
Depth 1 (Spawned)	44%	10%

A matrix of Depth and Intact/Split. The percentage of prefixes in each category is shown.

# Number of splitters and spawners is small...



6500 splitters and spawners contribute 80% of prefixes

# Measuring the Extent of Load Sharing

- We used a heuristic to distinguish load sharing prefixes and multihoming assignments at depth one:
  - Depth 1 prefixes with the **same** Origin AS as their Depth 0 parent are called **load shared prefixes (or LS prefixes)**. Here we are referring to **load sharing done by ISPs or end-users with direct allocations**.


Example of LS → 1.4.0.0/15    Origin AS = 700  
1.4.0.0/24    Origin AS = 700



## More about the heuristic...

- Depth 1 prefixes with an Origin AS **different** from the Depth 0 parent are called **multihoming assignments (or MA prefixes)**. Here we are referring to customers advertising multihoming assignments for backup connectivity, load sharing etc.

2.2.0.0/15    Origin AS = **700**

Example of MA  2.2.2.0/24    Origin AS = **863**

•Note: Just **1%** of the prefixes in the routing table have more than one Origin AS. Therefore, we ignore such prefixes and still remain accurate.

•The MA prefixes may contain some load sharing prefixes (ISP with multiple ASes). Thus they serve as an **upper bound**.



# Load sharing – largest chunk

Intact

Split

Depth0

8000

19000

(These are  
Load Sharing  
Too)

Depth1

33000

7000

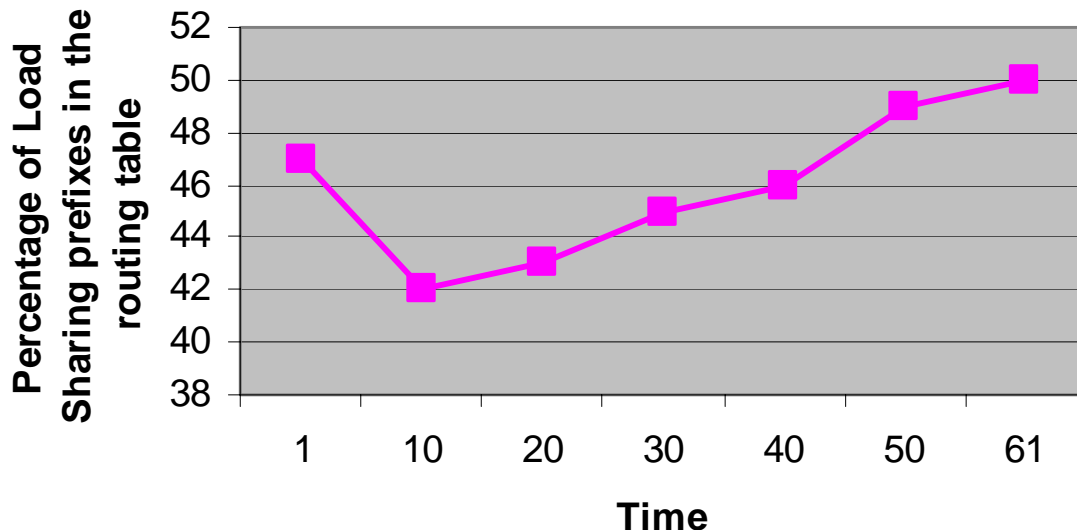
(21000 MA +  
12000 LS)

(4000 MA +  
3000 LS)

# Trend of the fraction of load sharing prefixes in the table

Over the last 5 years, the fraction of Load Sharing prefixes in the routing table is **increasing**

Number of Load Sharing prefixes is growing faster than the rest of the routing table



Month 1 must have seen some sort of BGP Misconfiguration?

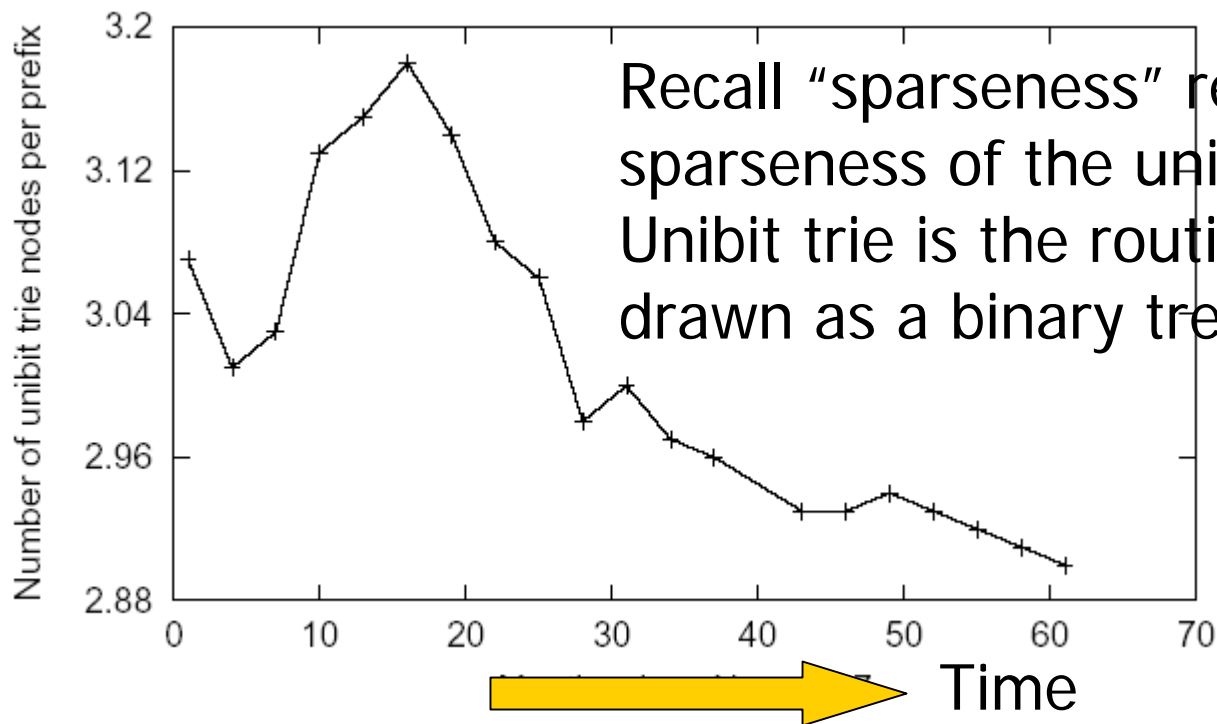


# Extent of Load Sharing

---

- The parameters Cspawn and Csplit can be used to measure load sharing.
- Data shows:
  - Average Cspawn over the last 5 years
  - MA            Constant (decreased slightly) (15%)
  - LS            Increasing (30% to 42%)
- Average Csplit increased from 48% to 69%.
- Conclusion: The extent of load sharing is steadily rising.

# Decreasing sparseness due to increasing load sharing



The size of the tree per prefix is decreasing. The values of  $C_{split}$  and  $C_{spawn}$  are increasing leading to prefixes being clustered together. MA prefixes are associated with low  $C_{spawn}$  values and are scattered over a much wider area.



# Can RIRs help reduce routing table size?

---

- RIRs try to reduce number of prefixes in the routing table
  - Aggregation of allocations is one of their goals. They reserve space for growth of current allocations.
- What if the RIRs renumber all networks I.e. give only one allocation per LIR?
- Not of much help.



# Why allocation practices can't help much...

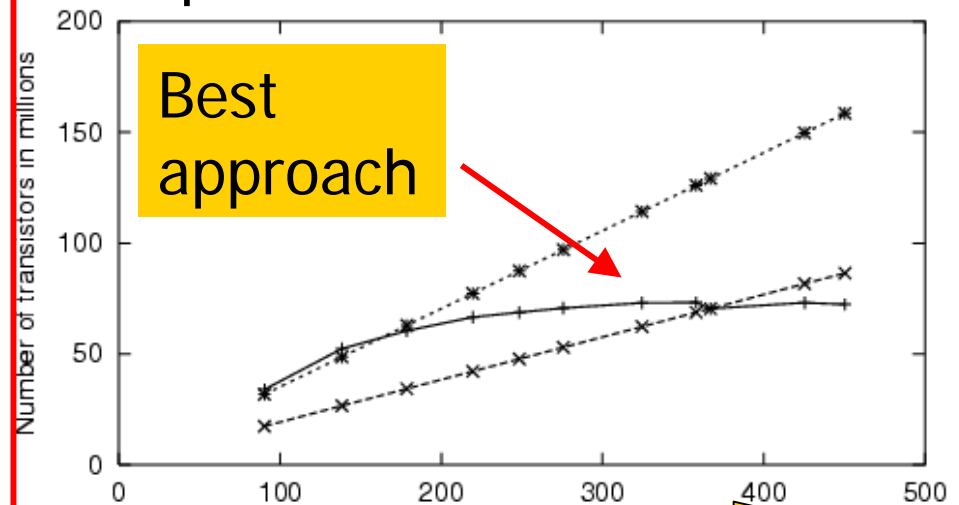
---

- This is because **only 11% of routing table prefixes are intact allocations.**
- The remaining 89% arise due to splitting and spawning – these are not under the control of RIRs.
- Conclusion:
  - Not much allocation practices can do to reduce routing table growth. Even renumbering would have a very small impact on routing table size.

# Conclusion

- The model can be used to predict routing tables based on allocation and routing practices.
- Load sharing a large and growing contributor to routing table growth.
- An application: model is useful for router vendors to calculate memory required to store forwarding table.

Forwarding table memory for different approaches when only Cspawn increases.



Routing table size increases  
Only due to increase in  
Cspawn



# Contact information

---

- Email: [hnarayan@cs.ucsd.edu](mailto:hnarayan@cs.ucsd.edu)