FIB Aggregation

Zartash Uzmi

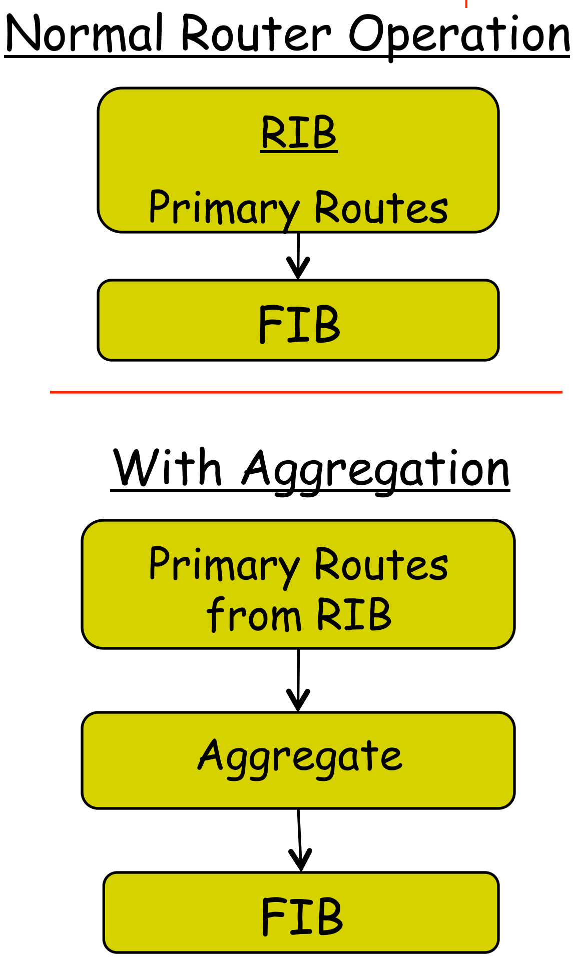
draft-uzmi-smalta-01

(with Ahsan Tariq and Paul Francis)

FIB Aggregation Work

- First introduced: IETF 76
  - draft-zhang-fibaggregation
  - Level 1-4

- SMALTA (at IETF78)
  - Better (near-optimal)



Changes since IETF 78

- Completed, but not reflected in current draft
  - Refinement of SMALTA
  - Thorough Evaluation (with data from a real ISP)
    - High confidence level in results

- In progress (Consolidation of the two drafts)
  - Original (Level 1-4) draft (IETF 76)
  - SMALTA draft (IETF 78)

Evaluation of SMALTA

- Data Sets
  - Routeviews (yearly: 12/2001 to 12/2010)
  - Various routers from a Tier-1 service provider
    - Based on router type, location, #interfaces

- Main findings: Savings
    - In FIB memory (line card): 35% and upwards (as large as 75%)
    - In #prefixes: ~12% better (than savings in memory)
    - In lookup time (#memory accesses): ~25% faster
    - Update processing: <1 FIB update per RIB update (on average)

L1/L2/SMALTA: Expectations?

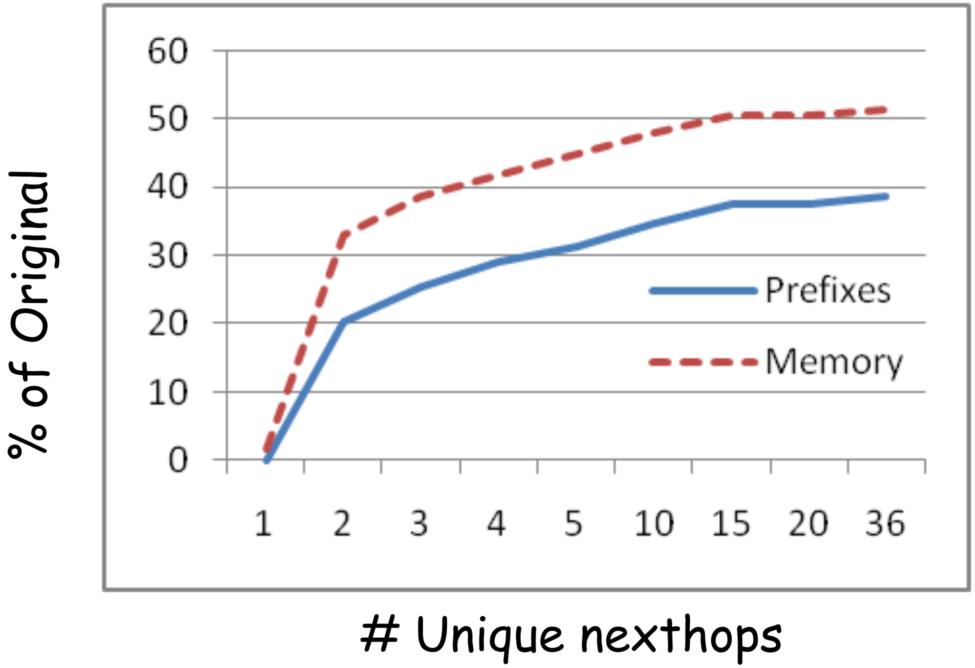
Aggregated prefixes (as % of original)

Router	SMALTA	Level 1	Level 2
R1	37%	68%	53%
R2	36%	66%	51%
R3	40%	68%	58%
R4	21%	55%	37%
R5	13%	49%	28%
R6	19%	54%	35%
R7	55%	79%	72%

For 2 Internet Gateway Routers (R1,R2) and 5 Access Routers in Provider Network

Aggregation and #next hops

Fewer aggregation opportunities with more nexthops

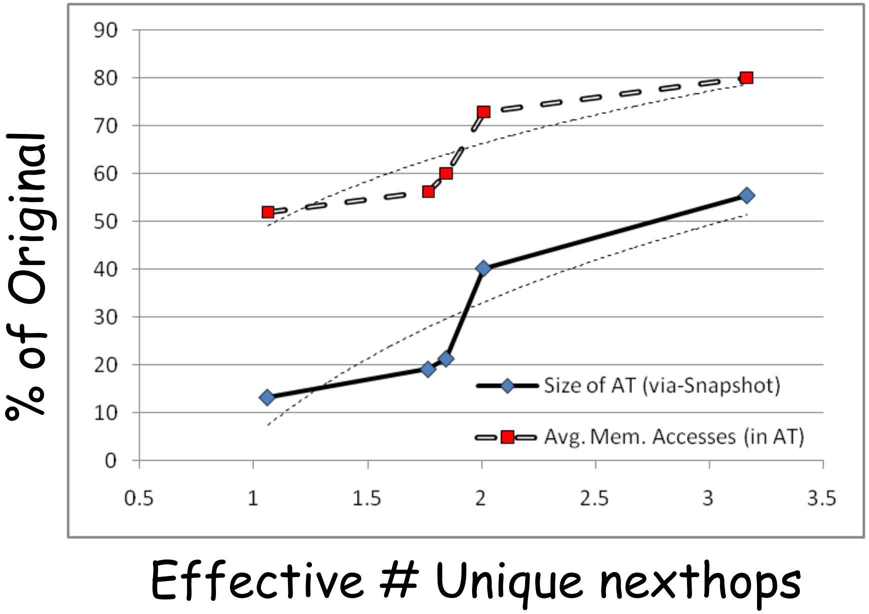


Routeviews
12/2010

Memory savings (for Tree Bitmap) are somewhat (~12%) lower

#Memory Accesses/Lookup time

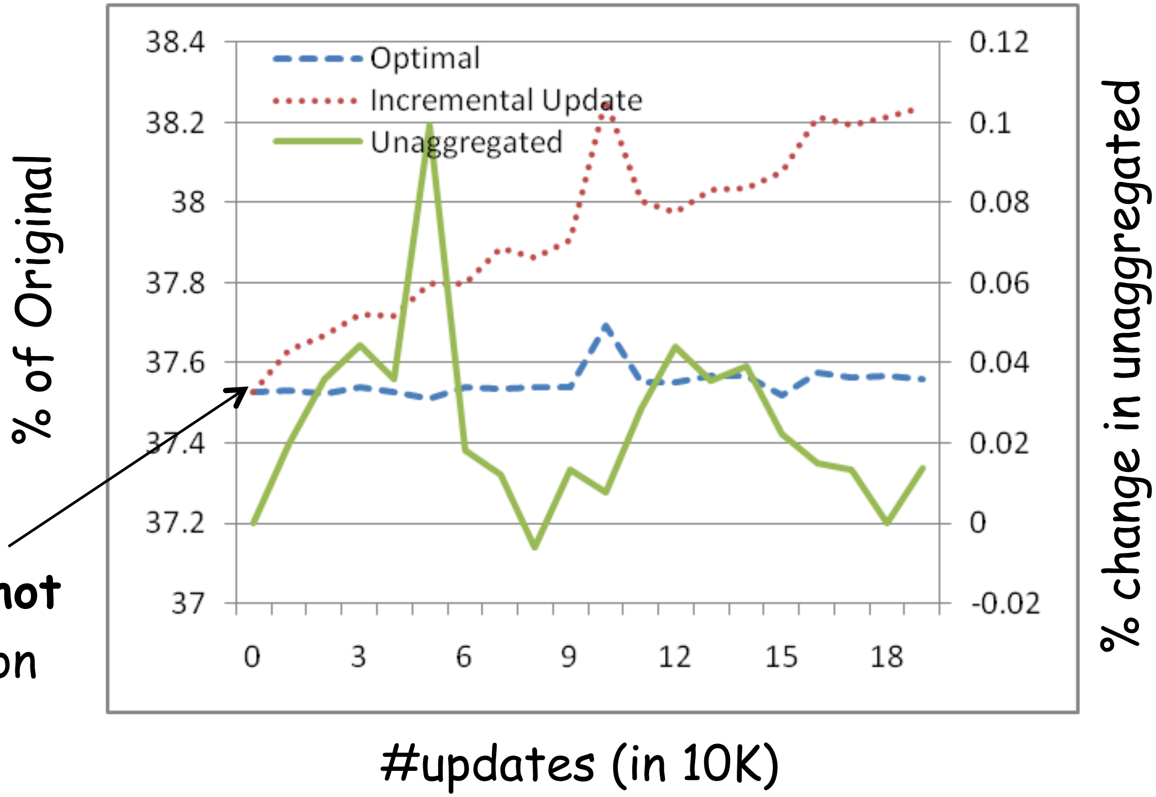
Lookup time (Tree Bitmap) varies in accordance with …
the #prefixes after aggregation



5 Access Routers
(Provider Network)

For Internet Gateway Routers, about 25% fewer memory accesses when using Tree Bitmap

Incorporating Updates



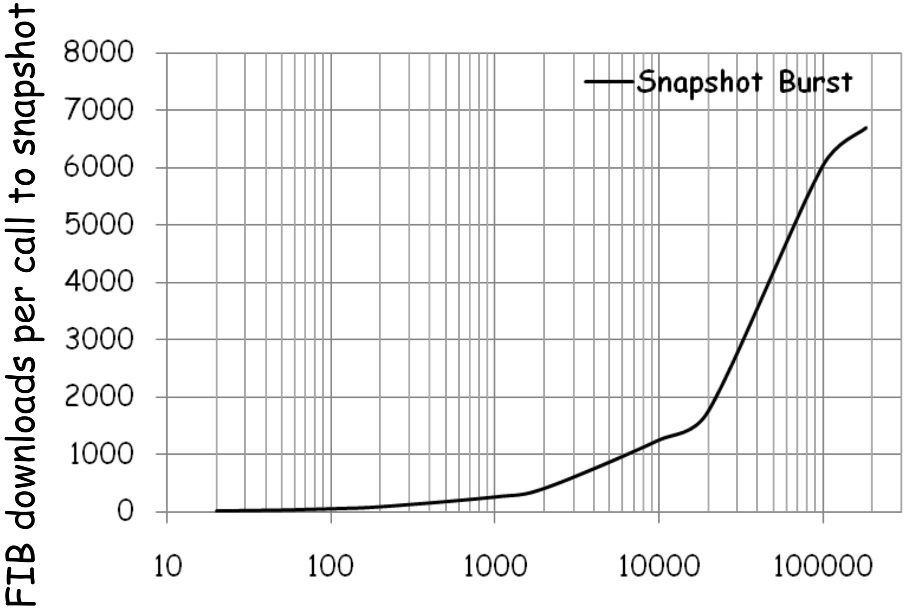
Call to **Snapshot** function

An IGR (Internet Gateway Router) in Provider Network

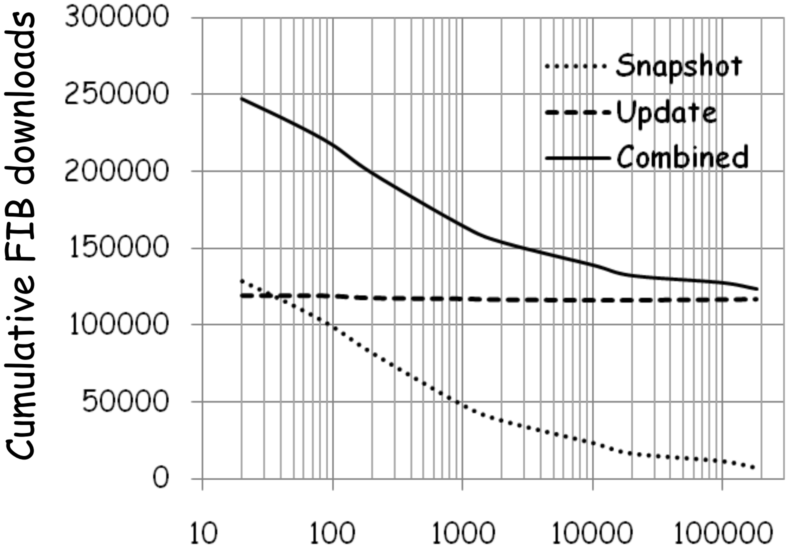
12-hour Update Trace

#aggregated prefixes is near-optimal after a large number of updates are incorporated

Updates → FIB downloads



#Updates b/w consecutive calls to "snapshot" function



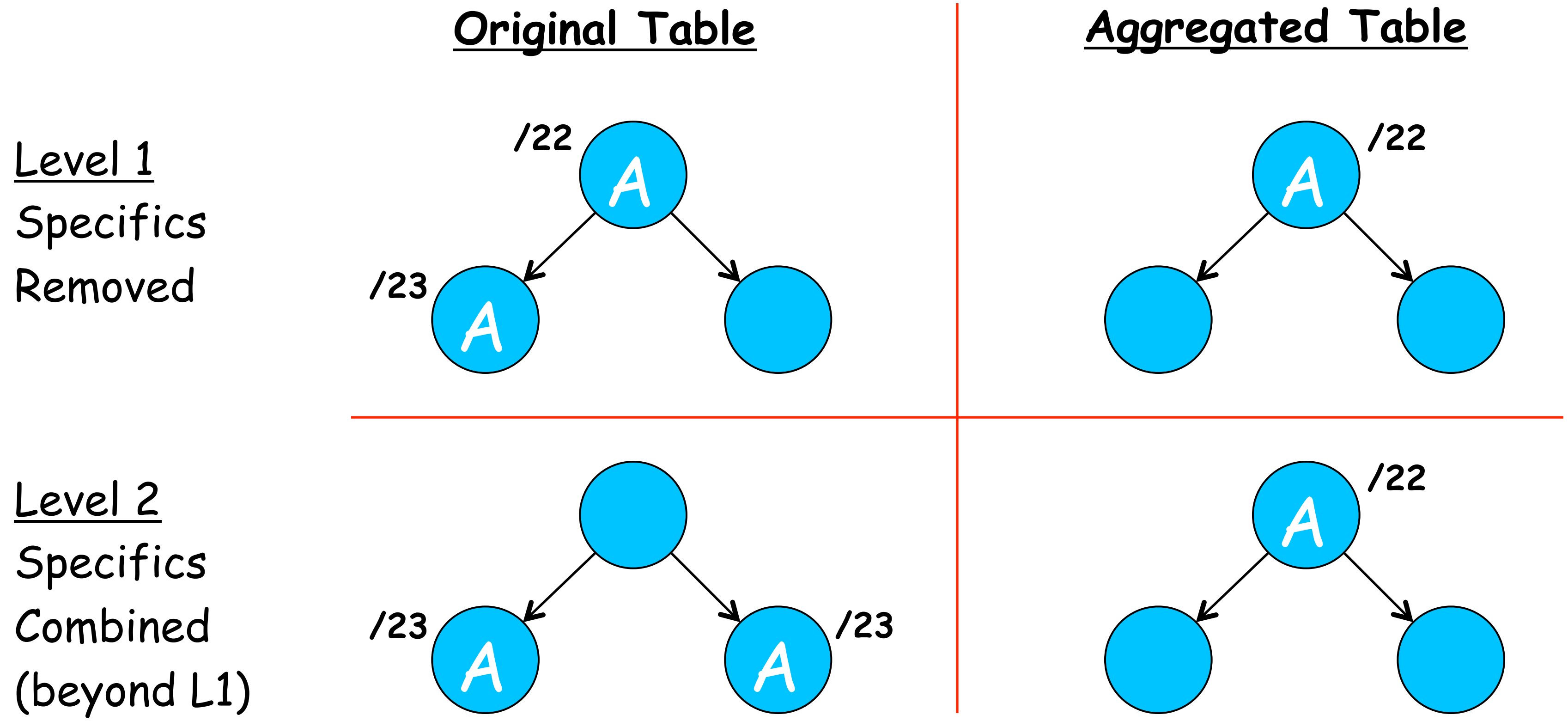
#Updates b/w consecutive calls to "snapshot" function

An IGR in Provider Network
12-hour Update Trace (~180K updates)

COMMENTS / QUESTIONS

ADDITIONAL SLIDES

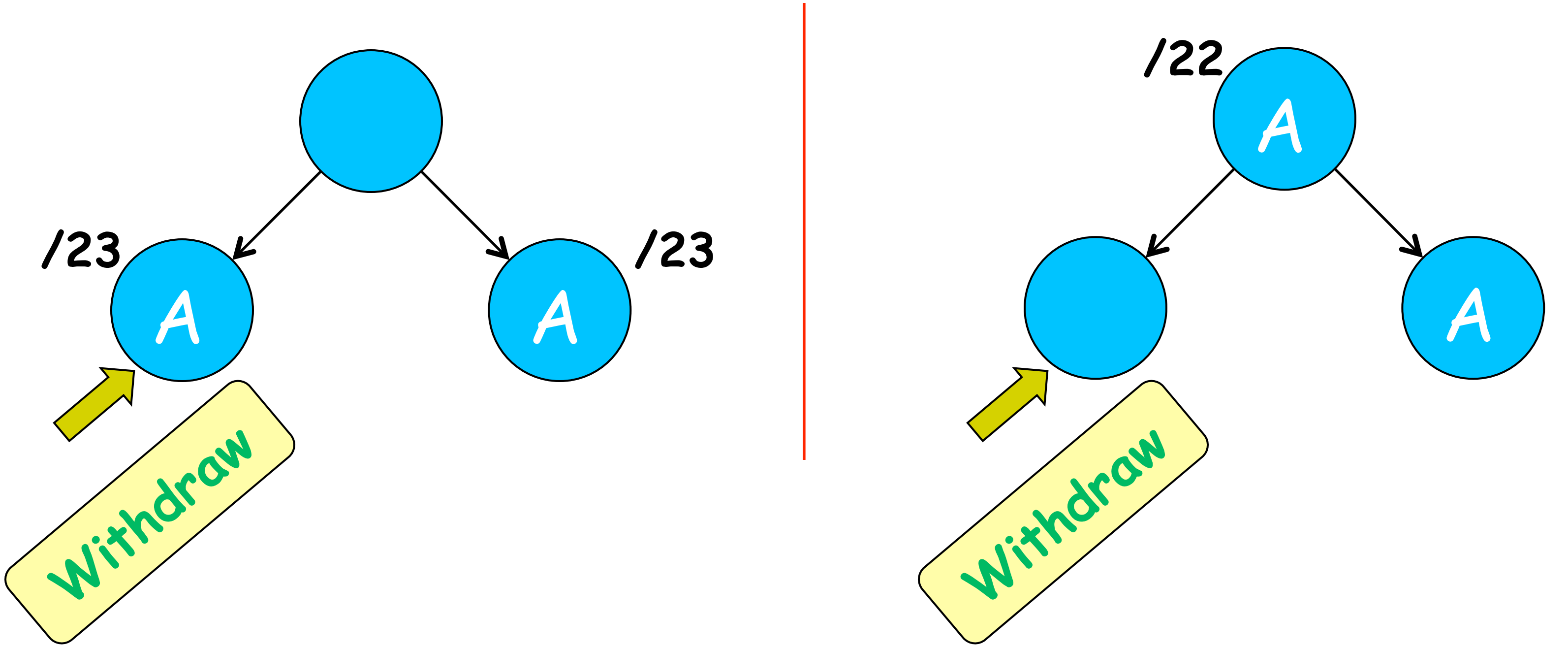
FIB Aggregation: basic idea



Exploit aggregation opportunities over entire Table

Basic Idea for Updates

Example 2: Aggregate specifics [having same next hop] – Level2



Where does SMALTA stand?

	Aggregation Opportunities	Updates	Whiteholing
Level 1	Specifics removed	Y	N
Level2	Specifics combined	Y	N
Level 3	Specifics combined over holes	Y	Y
Level 4		Y	Y
ORTC [1999]	Exploits all: Optimal	N	N
SMALTA	Exploits all (~ORTC)	Y	N

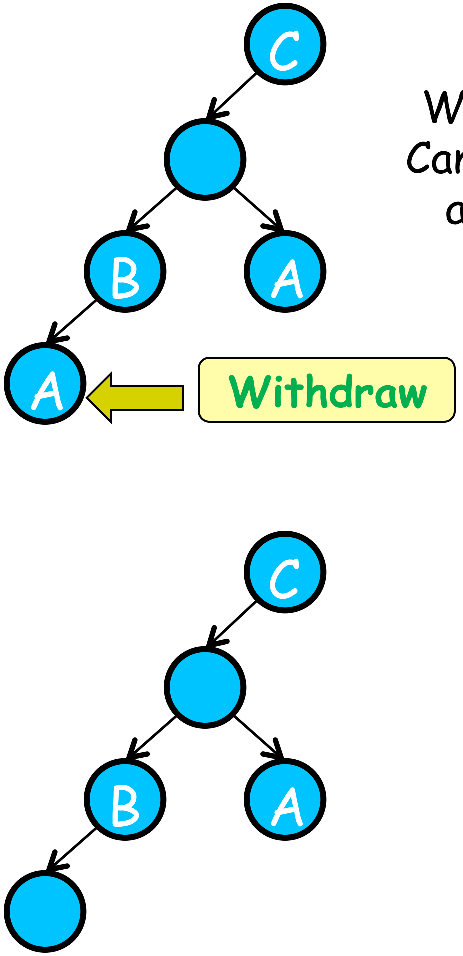
RIB snapshot → Aggregate → FIB: Snapshot Algo

BGP updates → Aggregated table: Update Algo

Snapshot and WITHDRAW

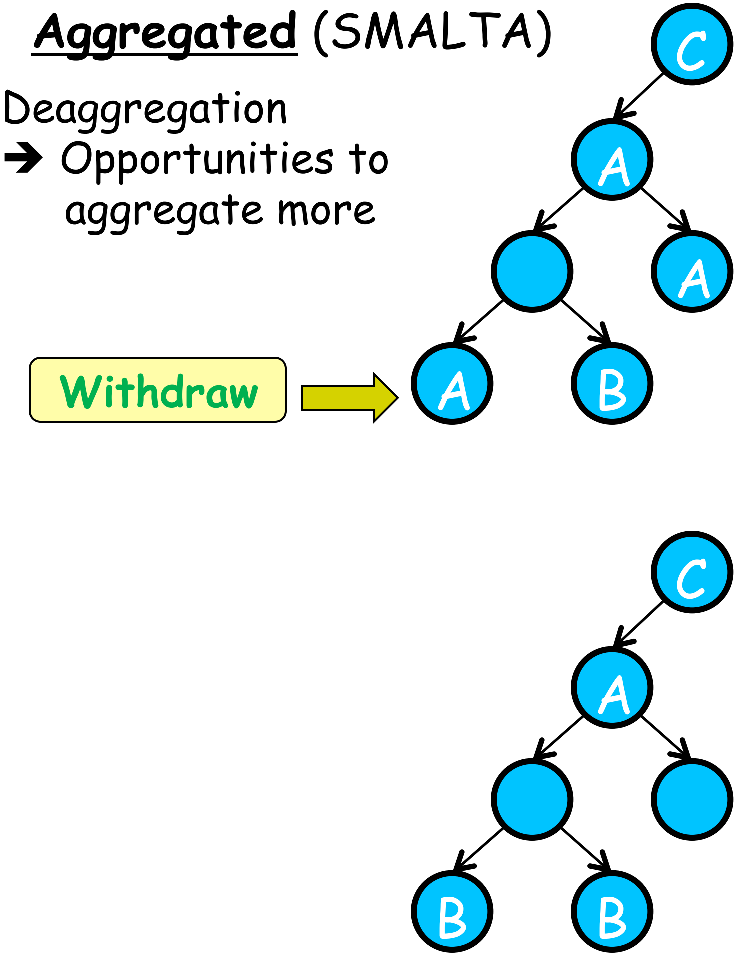
Original

With Level 1-4
Can't aggregate
any further!



Aggregated (SMALTA)

Deaggregation
➔ Opportunities to aggregate more



Remarks

- SMALTA Snapshot (300-400ms)

  ~3-4x more processing than L1 and L2

  Applied infrequently

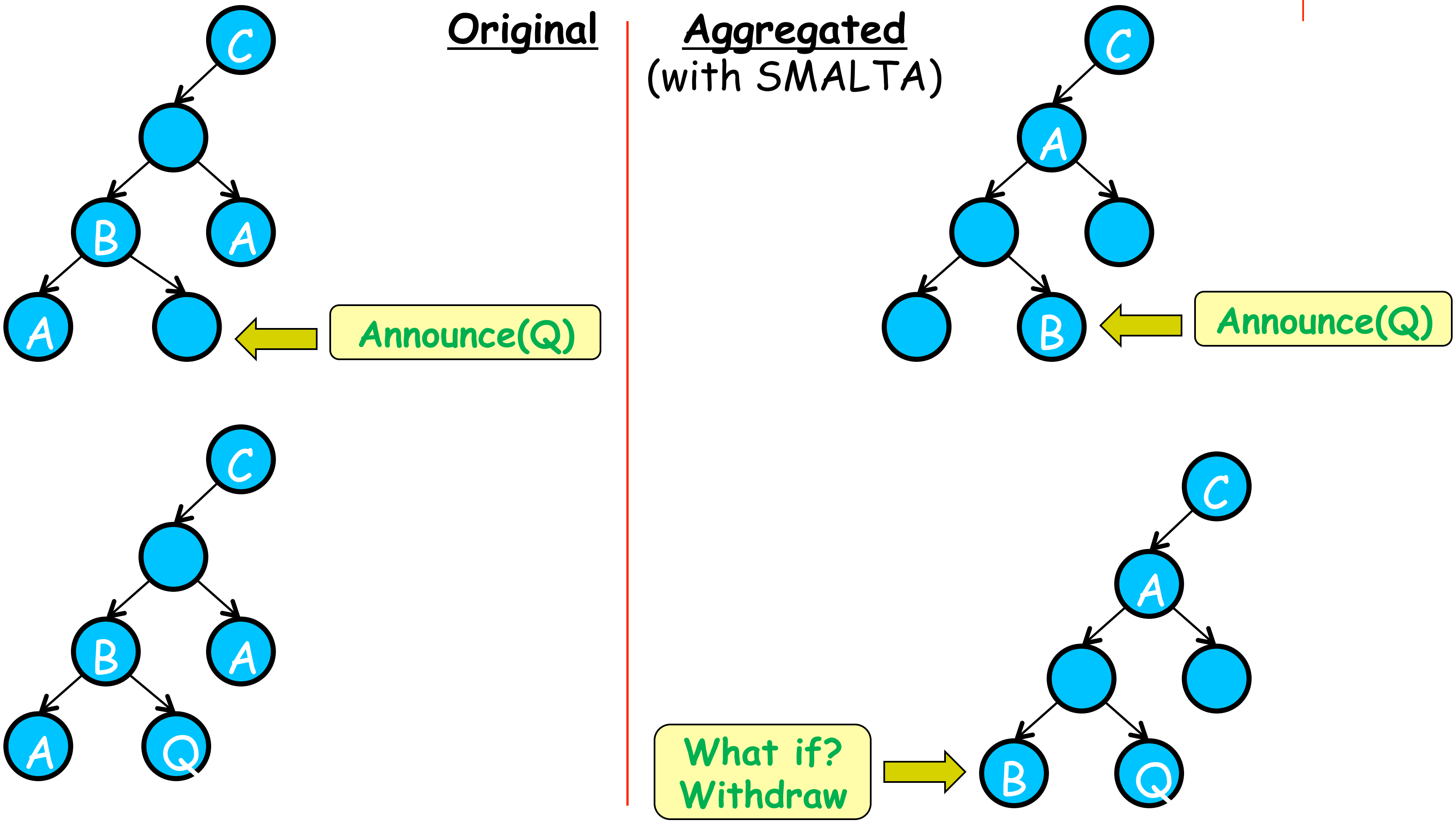
- SMALTA Update

  ~ same processing time as L1 and L2 (typical: 3μs)

  Fewer avg. RIB-to-FIB downloads

- Our view: another option for FIB aggregation

One-shot + ANNOUNCE + WITHDRAW



Original

Aggregated
(with SMALTA)

Announce(Q)

Announce(Q)

What if?
Withdraw

Incremental Updates: Analysis

- How far aggregated you are after N updates?
- How long does it take to incorporate updates?
- How many RIB to FIB downloads per update?

Practicalities

- Can't aggregate entire table on every update
    - Snapshot aggregation
        - Take current snapshot of RIB and Aggregate
        - On "significant" routing changes (e.g., BGP hard reset)
        - Perform a monolithic download after Snapshot
- To reflect BGP updates in FIB
    - Incremental updates to aggregated table