Measuring Large Traffic Aggregates on Commodity Switches

Lavanya Jose, Minlan Yu, Jennifer Rexford
Princeton University, NJ

Motivation

- Large traffic aggregates?
  - manage traffic efficiently
  - understand traffic structure
  - detect unusual activity

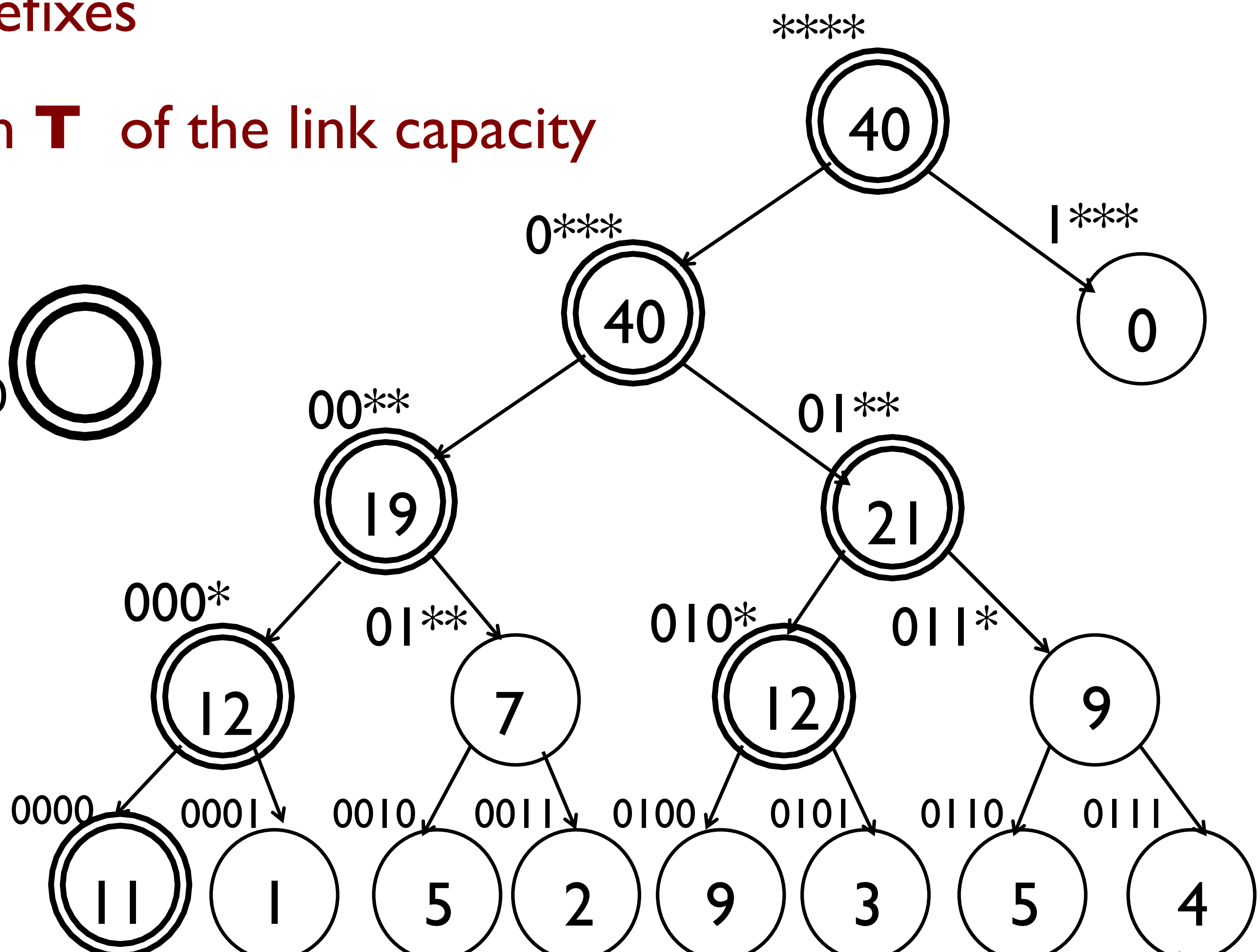
Aggregate at fixed prefix-length?

- Top 10 /24 prefixes (by how much traffic they send)
  - could miss individual heavy users
- Top 10 IP addresses …
  - could miss heavy subnets where each individual user is small

Aggregate at all prefix-lengths? (Heavy Hitters)

- All the IP prefixes
- >= a fraction **T** of the link capacity

HH: sends more than
T= 10% of link cap. 100

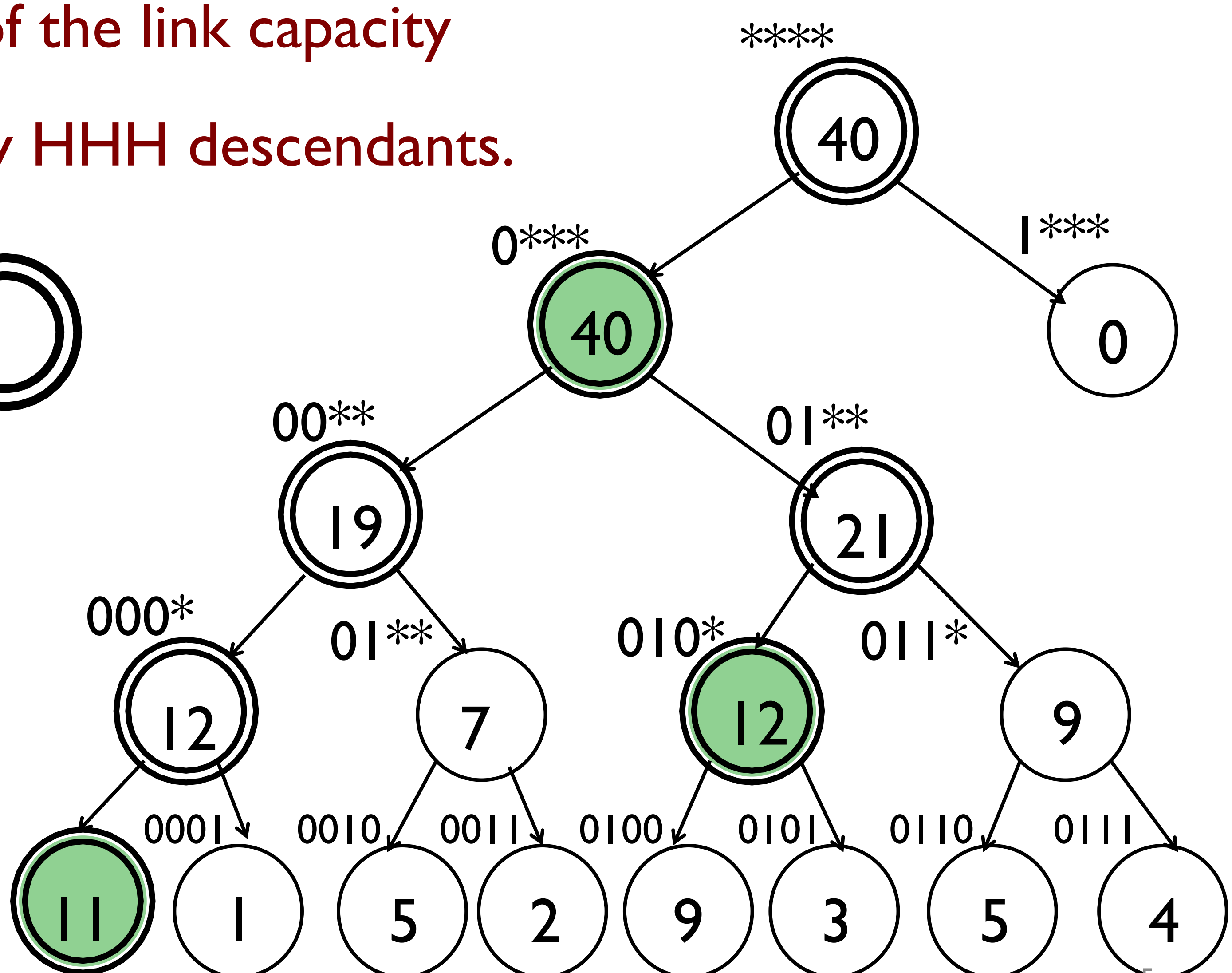


Hierarchical Heavy Hitters

- All the IP prefixes
- >= a fraction **T** of the link capacity
- after excluding any HHH descendants.

HH: sends more than T= 10% of link cap. 100

HHH:



Related Work

- Offline analysis on raw packet trace [AutoFocus]
  - accurate but *slow* and *expensive*
- Streaming algorithms on Custom Hardware [Cormode'08, Bandi'07, Zhang'04, Sketch-Based]
  - accurate, fast but *not commodity*

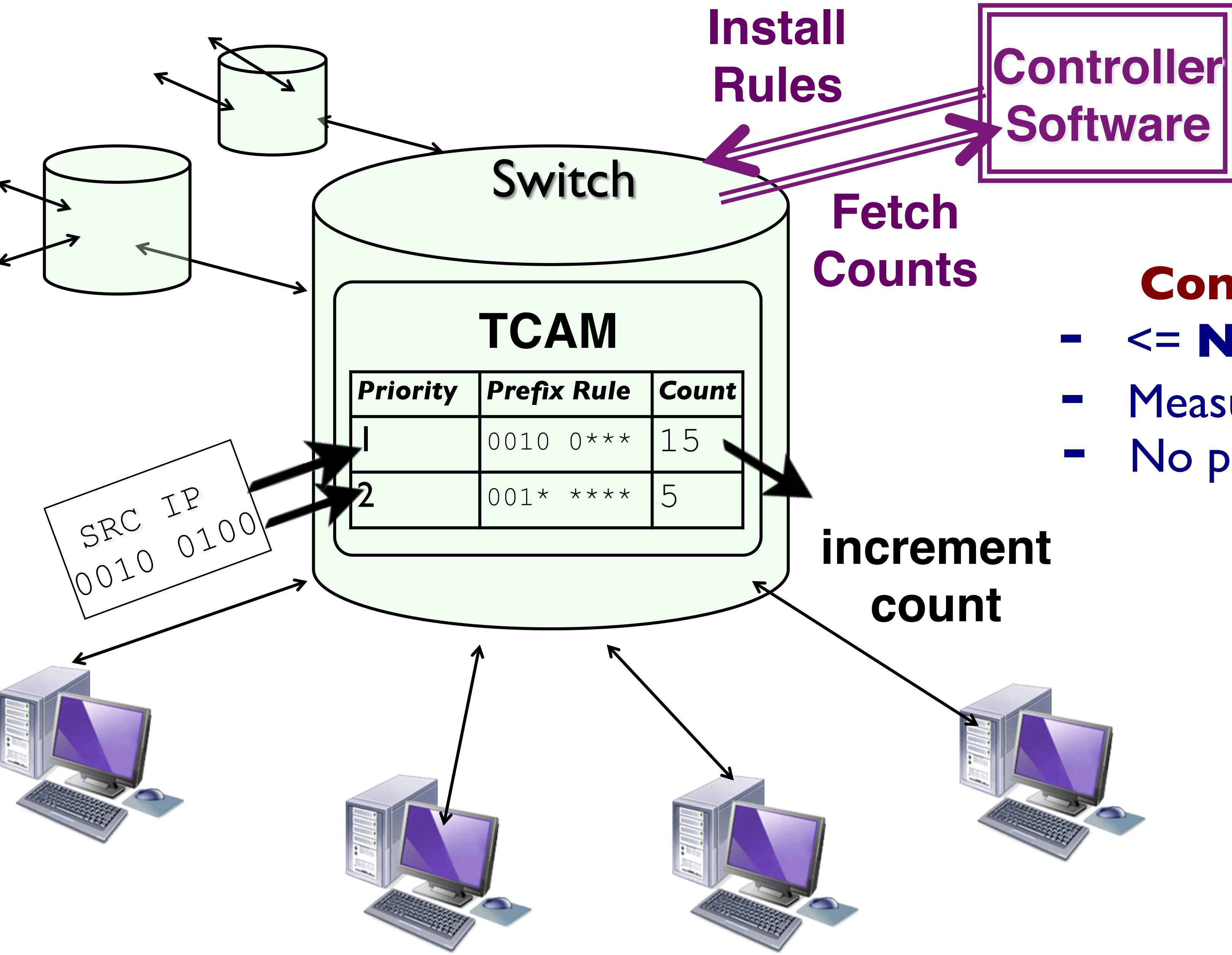
Our Work:
Commodity, fast and relatively accurate

HHH on Commodity- Using OpenFlow

- Why commodity switches?
  - cheap, easy to deploy
  - let "network elements monitor themselves"
- Commodity OpenFlow switches
  - available from multiple vendors (HP, NEC, and Quanta)
  - deployed in campuses, backbone networks
  - wildcard rules with counters to measure traffic

Priority	***Prefix Rule***	***Count***
1	0010 0*** ...	15
2	001* **** ...	5

OpenFlow Measurement Framework



Constraints

- <= **N** Prefix Rules
- Measuring Interval **M**
- No pkts to Controller

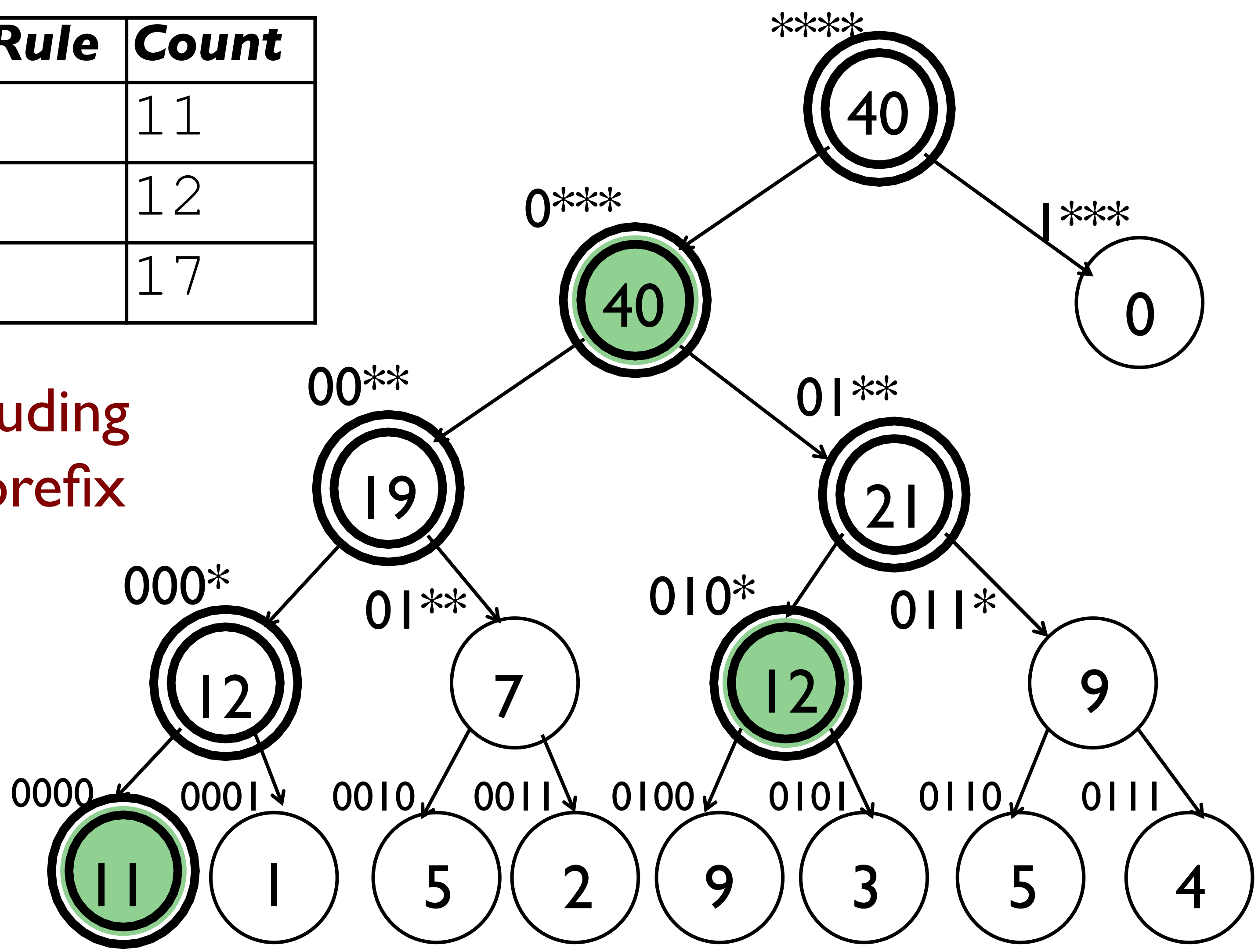
Monitoring HHHes

Priority	***Prefix Rule***	***Count***
1	0000	11
2	010*	12
3	0***	17

HHH: after excluding any descendant prefix rules

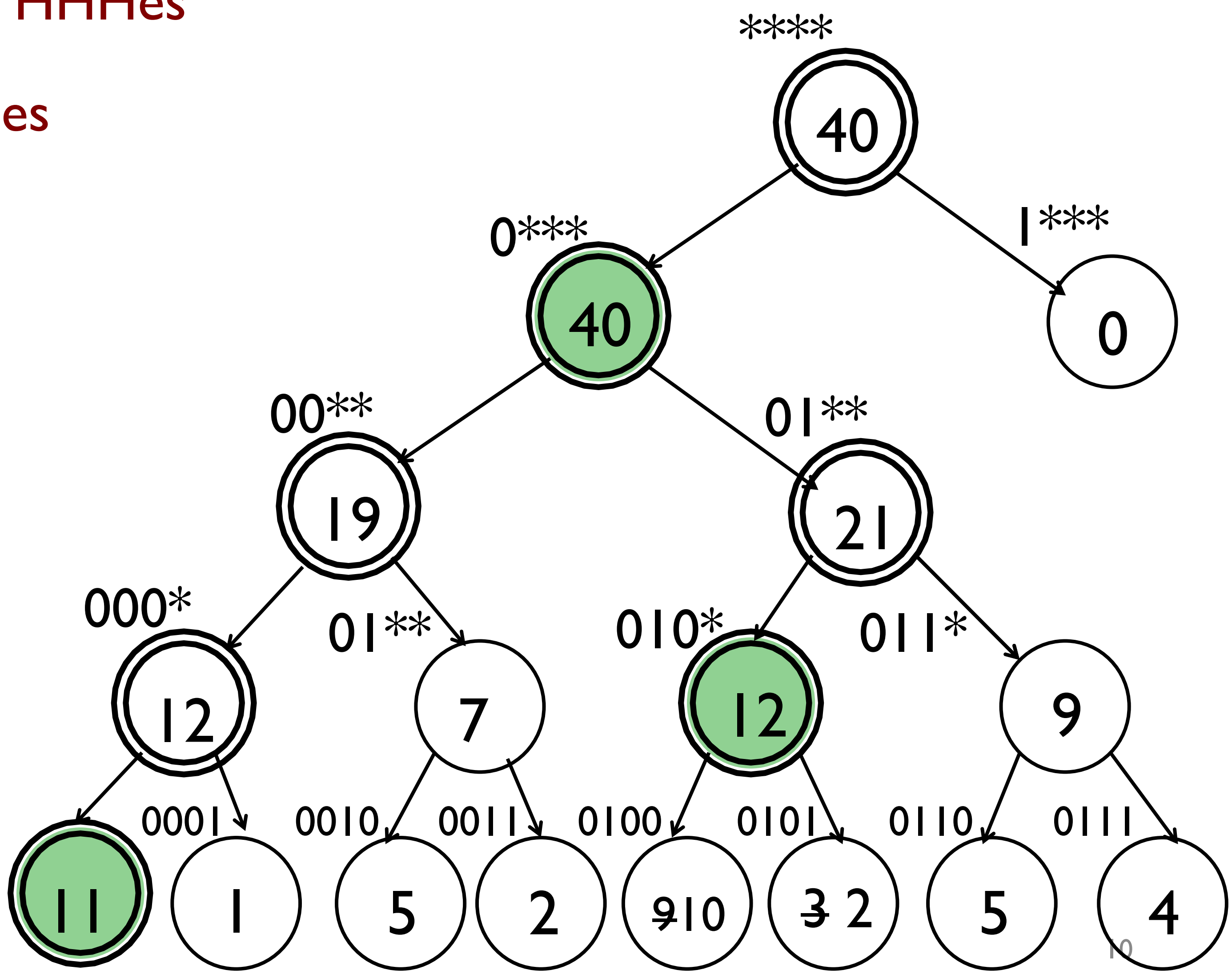
TCAM: priority matching

A perfect match!



Detecting New HHHes

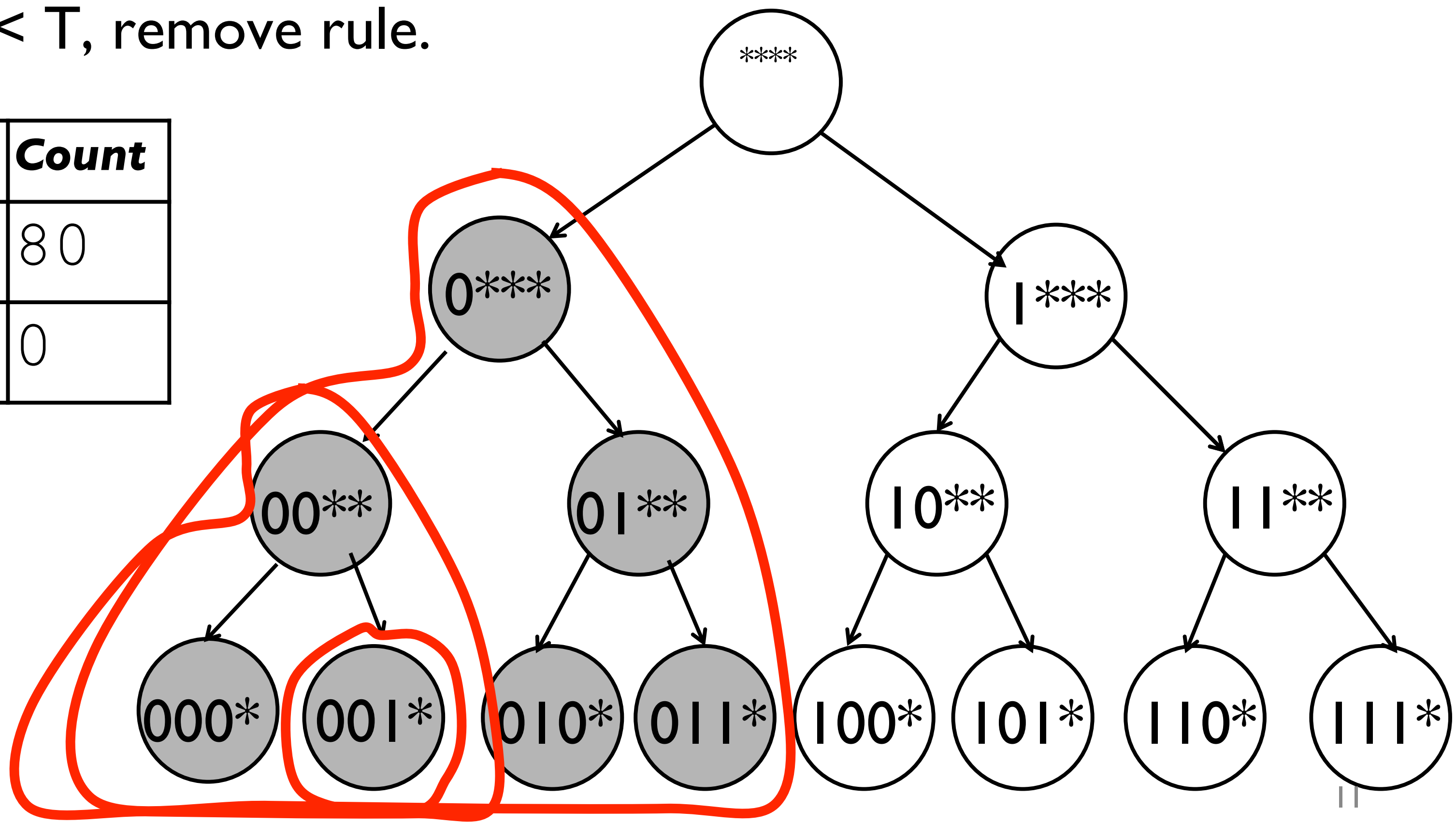
- Monitor children of HHHes
- Use at most 2/T rules



Identifying New HHHes

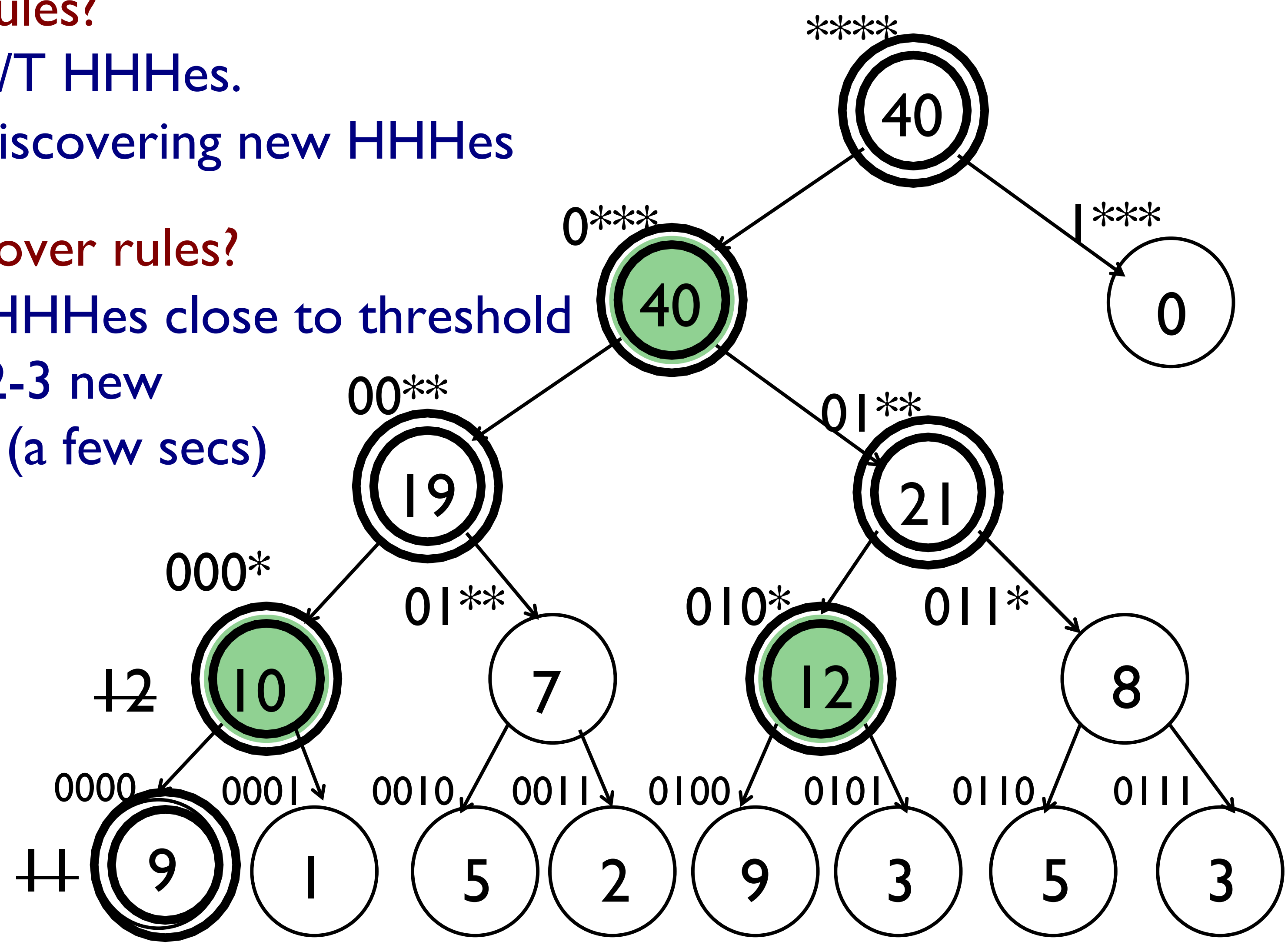
- Iteratively adjust wildcard rules:
  - Expand
    - If count > T, install rule for child instead.
  - Collapse
    - If count < T, remove rule.

Priority	*Prefix Rule*	*Count*
1	0***	80
2	****	0



Using **Leftover** Rules

- Why left over rules?
  - May not be 1/T HHHes.
  - May still be discovering new HHHes
- How to use leftover rules?
  - To monitor HHHes close to threshold
  - Data shows 2-3 new HHHes/ interval (a few secs)

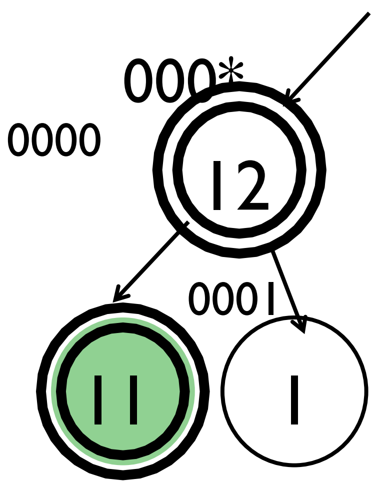


Evaluation- Method

- Real packet trace (400K pkts/ sec) from CAIDA
  - Measured HHHes for T=5% and T=10%
  - Measuring interval M from 1-60s

Evaluation- Results

- 20 rules to identify 88-94% of the 10%- HHHes
- **Accurate**
    - Gets ~9 out of 10 HHHes
    - Uses left over TCAM space to quickly find HHHes
    - *Large traffic aggregates usually stable*
- **Fast**
    - Takes a few intervals for 1-2 new HHHes
    - Meanwhile aggregates at coarse levels



Stepping back… not just for HHHes

- Framework
  - Adjusting <= N wildcard rules
  - Every measuring interval M
  - Only match and increment per packet
- Can solve problems that require
  - Understanding a baseline of normal traffic
  - Quickly pinpointing large traffic aggregates

Conclusion

- Solving HHH problem with OpenFlow
  - Relatively accurate, Fast, Low overhead
  - Algorithm with expanding /collapsing
- Future work
  - multidimensional HHH
  - Generic framework for measurement
    - Explore algorithms for DoS, large traffic changes etc.
    - Understand overhead
    - Combine results from different switches