Identifying BGP Routing Table Transfer

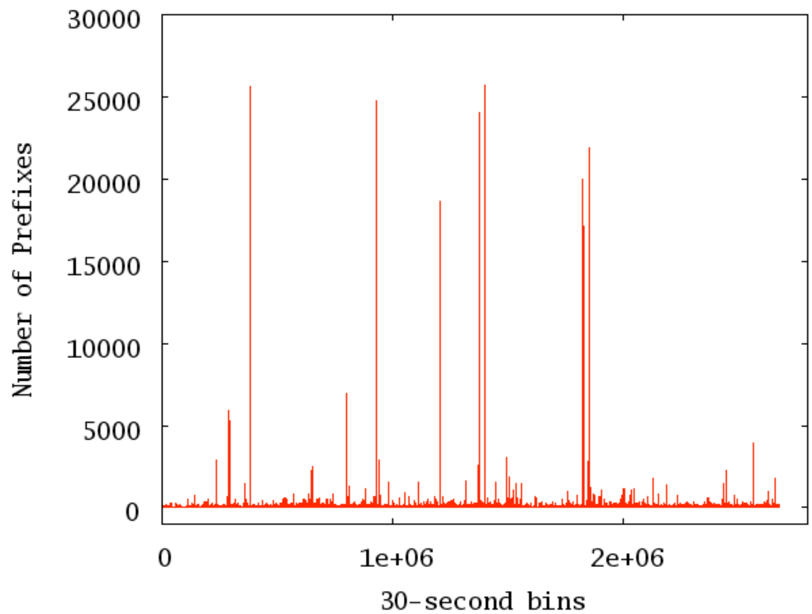
Beichuan Zhang (Univ. Of Arizona)
Vamsi Kambhampati (Colorado State Univ.)
Daniel Massey (Colorado State Univ.)
Mohit Lad (Univ. Of California, LA)
Lixia Zhang (Univ. Of California, LA)

Tool URL: http://netsec.cs.colostate.edu

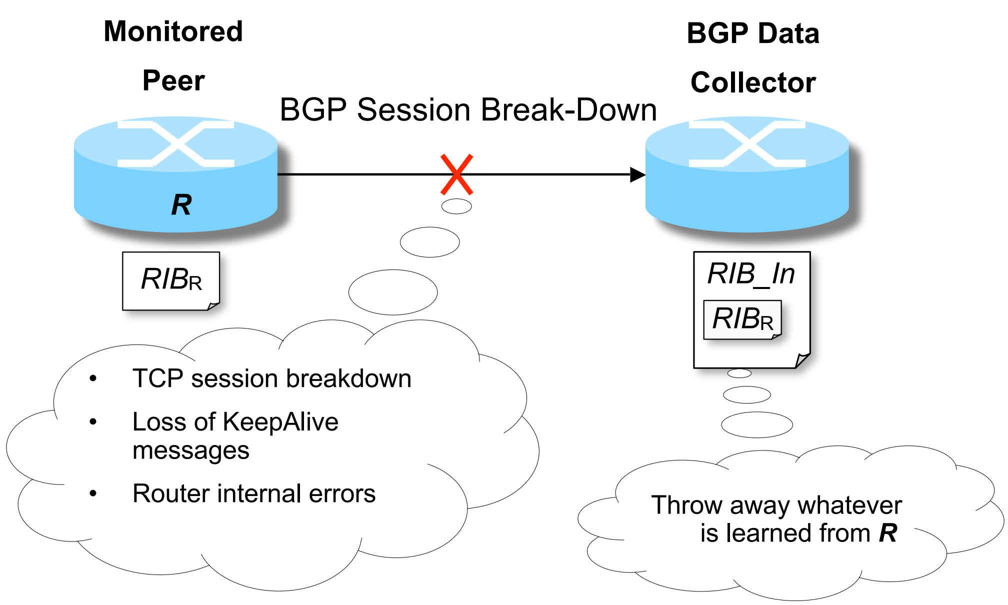
!Machu Picchu!

http://www.culturefocus.com/machu-picchu-1.htm

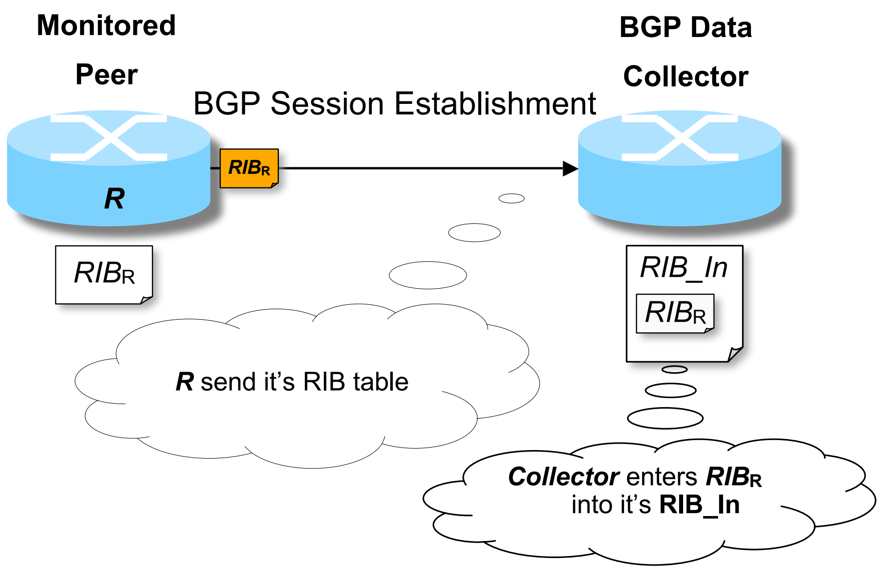
BGP Table Transfers



Session Resets & Table Transfers



Session Resets & Table Transfers



Why Identify Table Transfers?

- Remove Data Collection Artifacts
  - Table Transfers don't reflect routing dynamics
  - First stage of most BGP analysis
- Understand How Often Resets Occur
  - Nearly 15 - 20% of updates due to table transfers!
  - Optimize BGP differently if problem is frequent resets instead of frequent incremental updates
- Prior approaches relied on ad-hoc techniques
  - Delete updates if # of update > 1000 in 30 seconds
  - Delete all ***duplicate*** announcements

Our approach is show to be accurate

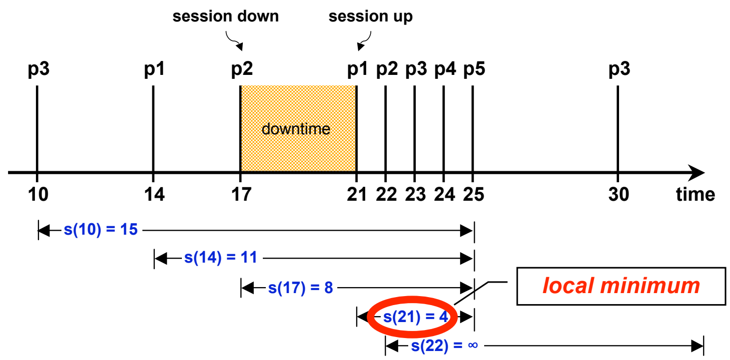
Table Transfer Problem

- Objective: Detect ***all*** table transfers
- For each table transfer, identify:
  - ***starting time***: timestamp of first table transfer update
  - ***duration***: how long the transfer lasted

Minimum Collection Time Algorithm

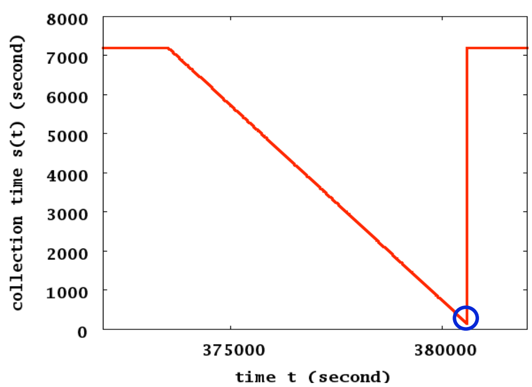
Collection Time

For an update received at time ***t***, define it's ***collection time***, ***s***(***t***), as the time it takes for *all* (*unique*) *prefixes to be announced*



Update Stream and Collection Time

Basic Approach



Sample $s(t) \sim t$

Basic MCT Algorithm:

1. For each update at time ***t***, calculate its collection time, ***s***(***t***)
2. Find all local minima of ***s***(***t***)
3. Each local minimum is considered as a table transfer.
    1. ***starting time*** is ***t*** (timestamp)
    2. ***duration*** is ***s***(***t***) (collection time)

Practical Tune-Ups

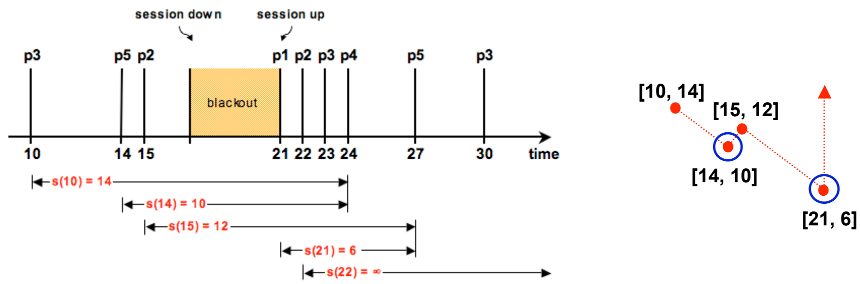
- Expected Table Size Estimation
- Conflicting Table Transfers
- Other practical tune-ups in paper

Expected Table Size

- Get Table Size from latest RIB snapshot
  - RIB as close as possible to updates
- Table Size may vary during downtime!
  - Incremental updates change table size too..
- Don't count all prefixes from table size
  - Instead, use a fraction ***N*** of last know table size
    - Where ***N*** = 0.99 (99%)

Conflicting Table Transfers

- Monotonicity breaks due to update timing!



- s(14), s(21) are two *conflicting* local minima!
  - A table transfer started (at t=21) when another table transfer is in progress (from t=14, to 14+10 = 24)

- Assume table transfers take short duration to complete
  - S(14) = 10 > 2(21) = 6, throw away s(14) and keep s(21)

Results

- Validation using RIPE Session Reset logs

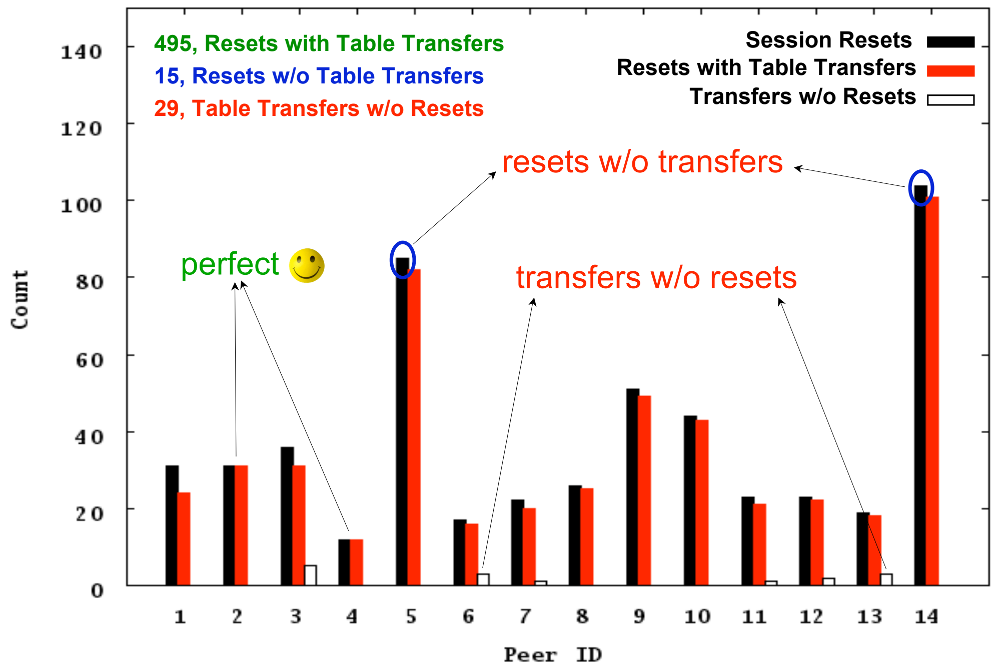
- Detection Accuracy
  - Does our approach find the exact starting time?

- Table Transfers in RouteViews Data
  Note: RouteViews does not provide session state logs

Tool Download URL:

http://netsec.cs.colostate.edu

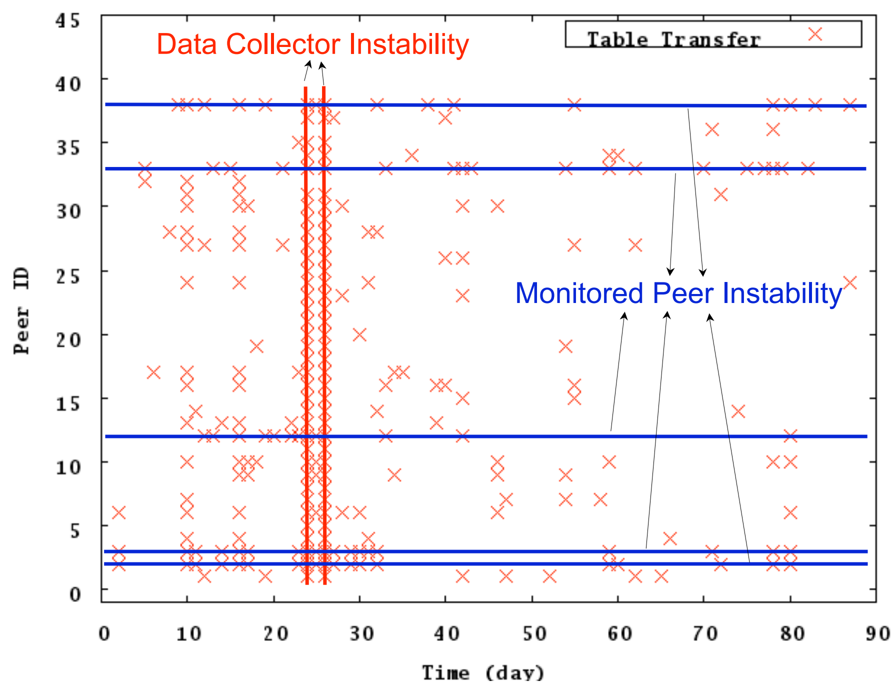
Validation Using RIPE Data



Detection Accuracy

- Inaccuracy (Offset)
  - Difference of:
    - Detected transfer start time
    - First update after reset log
- *Zero* Offset for 90% of Table Transfers
- For those with Non-Zero Offset
  - 50% are below 30 seconds
  - 50% have only 1 update (within offset time)

Table Transfers in RV Data



Comparison With Prior Approaches

- Duplicate removal approach
  - Remove ***all duplicate*** announcements
- 30 sec bin based approach
  - Remove if # of updates > 1000 in 30 seconds

- Both remove table transfer updates
- But .. both remove a lot of valid updates!

- Our Approach identifies table transfers accurately

Conclusion & Future Work

- Download our tool and see for yourself
  http://netsec.cs.colostate.edu 

- Future Work:
  - Identify table transfers due to remote resets

Questions?