Detecting Behavior Propagation in BGP Trace Data

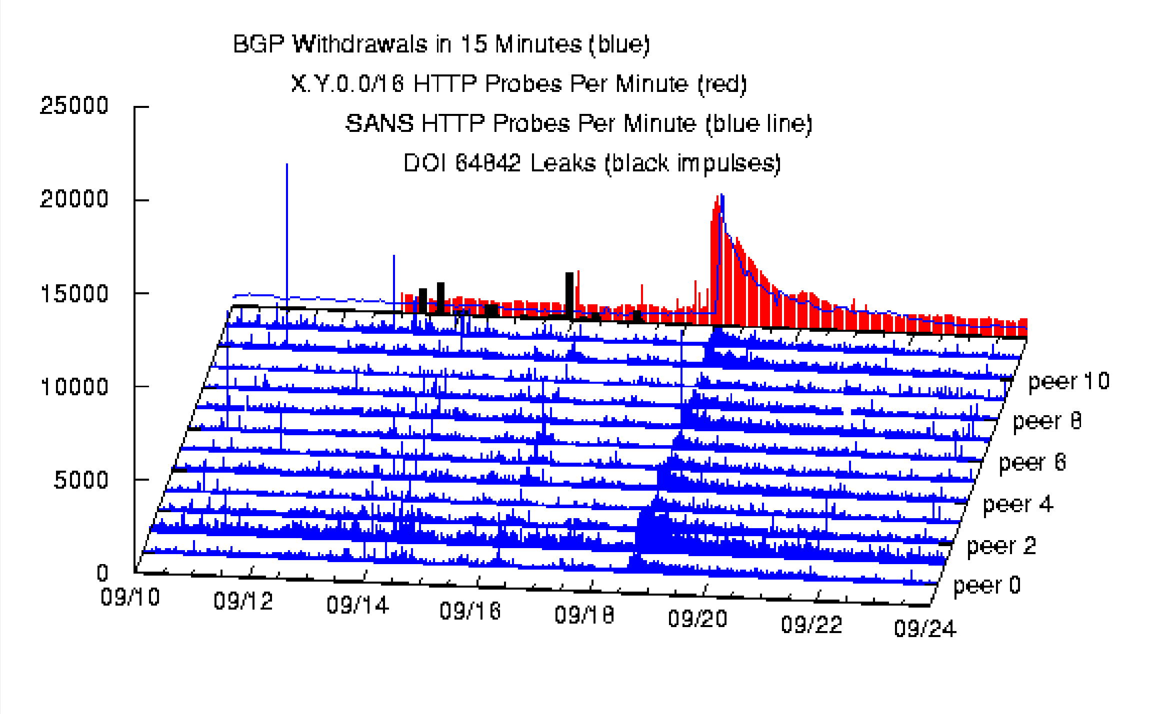
Brian J. Premore
Michael Liljenstam
David Nicol

Institute for Security Technology Studies,
Dartmouth College

Motivation

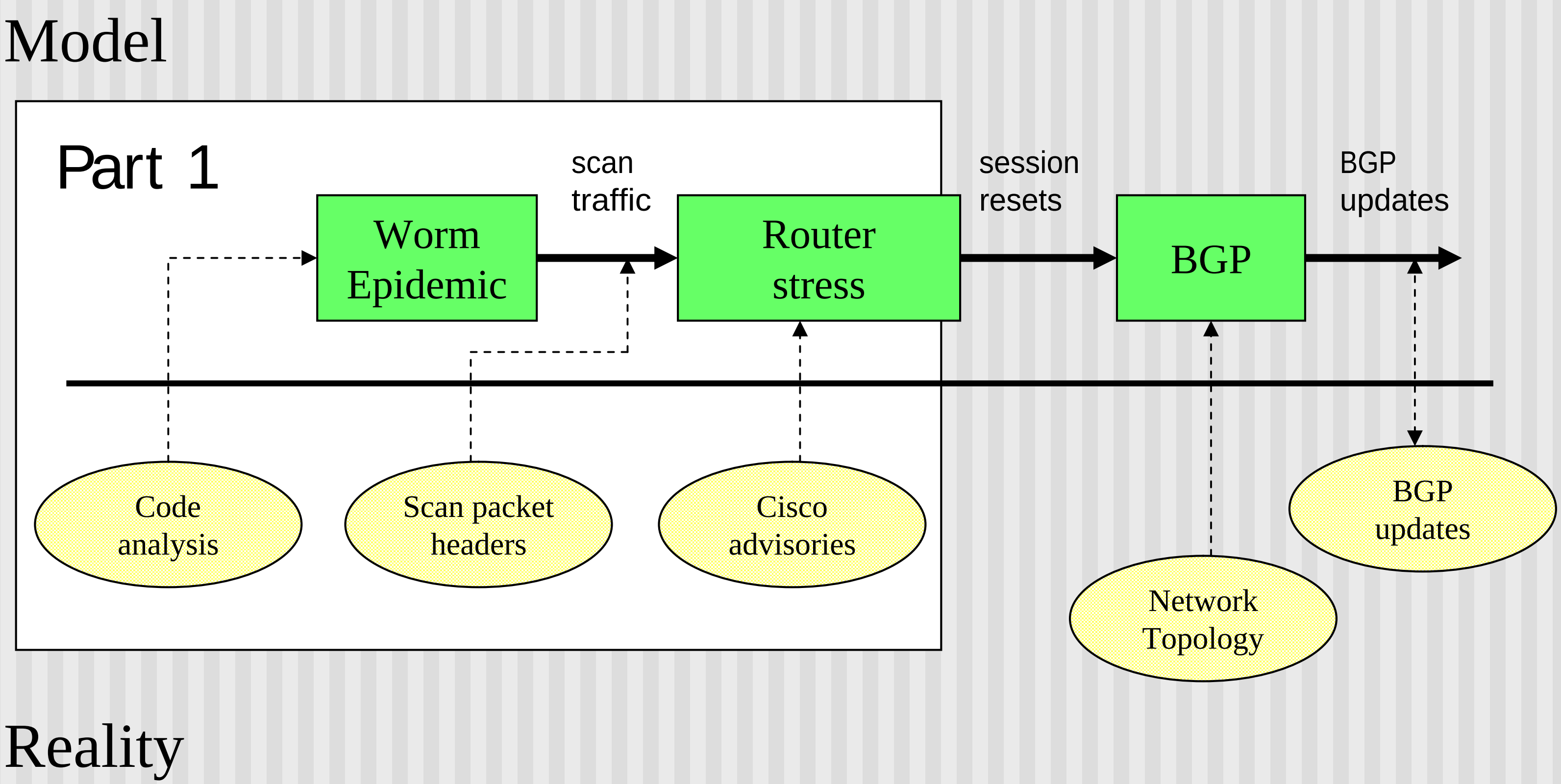
Is there a causal connection between large-scale worm infestations and BGP update message surges?

- Observed correlation [Cowie et al., '02]
  - Globally visible BGP update bursts
  - Correlated with Code Red v2 & Nimda



Motivation

- Use simulation to help answer…



Part 1: From Worm to Scans

- Relying on related work on worm studies
  - Moore, “Code-Red: a case study on the spread and victims of an Internet worm”, IMW’02
  - Staniford et al., “How to 0wn the Internet in Your Spare Time”, USENIX Security ’02
  - And numerous security advisories, code analysis reports, etc.

Part 1: From Worm to Scans

Work on Modeling/Simulation:

- **“A Mixed Abstraction Level Simulation Model of Large-Scale Worm Infestations”,**
  to be presented at MASCOTS’02 Symposium

Key issues addressed:

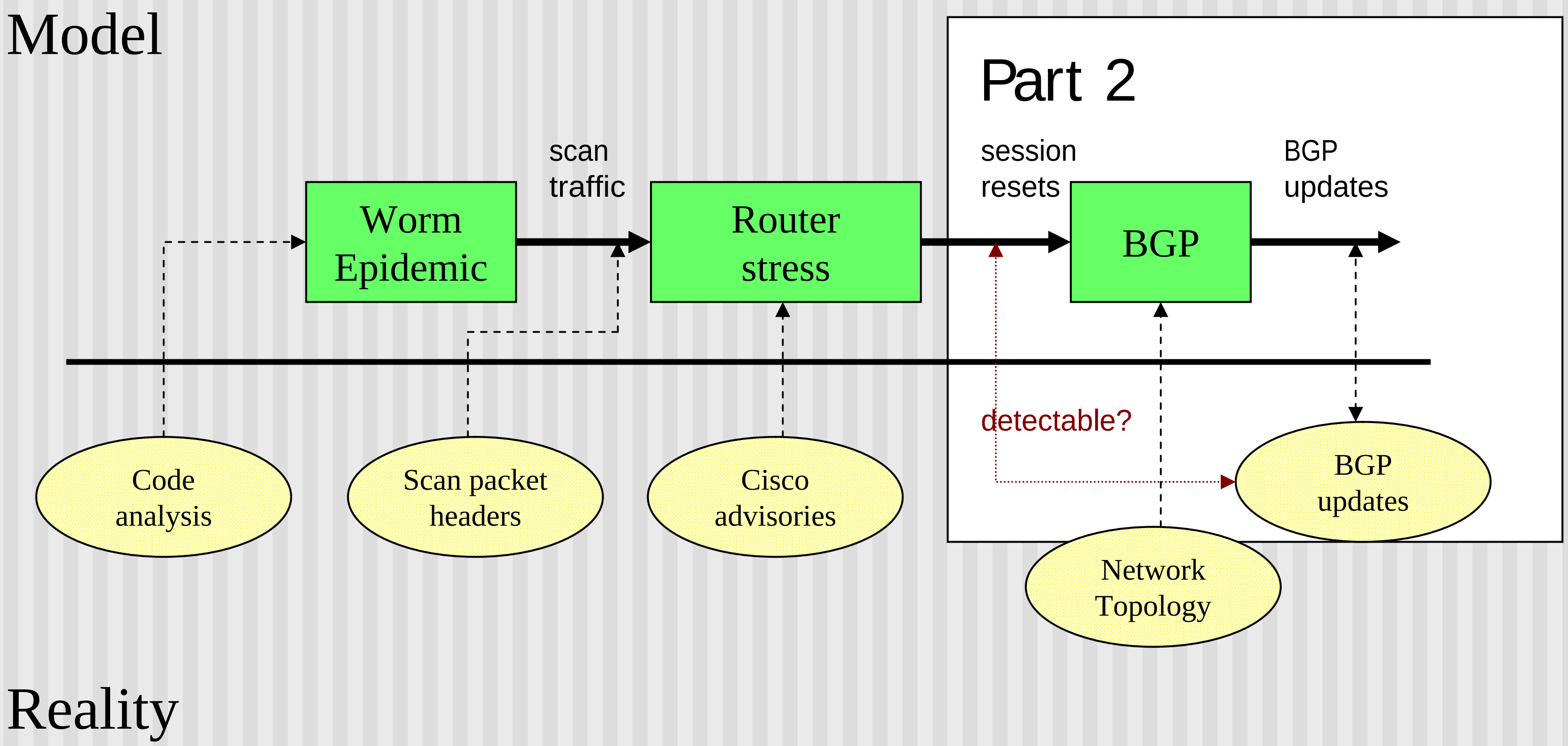
- How to efficiently simulate a model with both
  - Worm
  - Infrastructure detail

⇒ develop/investigate:

  - Epidemic models
  - Memory constraints and model scalability

Current Work

Part 2: Effects of BGP ⇒ Back to data



Questions

- Is it possible to detect traces of (remote) sources of instability, including session resets, from the BGP update data?
- If so, is there a significant increase in resets during the worm events that could indicate causal effects from worm?
- If so, *where* were these occuring? In large transit ASes, or small edge ASes?
  - This could give us clues for causal link conjectures to model…

Sneak Preview of Coming Attractions

- Early attempts at detecting BGP session resets
  - Using the “BGP RTG” tool, [Maennel and Feldman]
- Filtering collection point Peer OPENs
  - Eliminating measurement artifacts
- Current efforts
  - Using “per AS update bursts”
  - Look for AS pair drop-outs
- Summary / Conclusions

The “BGP RTG” Tool

- BGP update message analysis tool developed at Saarland University
- Includes heuristic for detecting (remote) BGP session resets
- Described in
  “Realistic BGP Traffic for Test Labs”, SIGCOMM’02

- *Could we use it to detect and locate hypothesized session resets (and router crashes) in the data?*

BGP RTG: Reset Heuristic

Session reset heuristic

- Look at each individual prefix update
- Move a 6 minute sliding window over the updates
- If a “large” fraction of the prefixes originating or transiting by an AS have been updated within the window this indicates a session reset, and these updates/ASes are marked as part of a reset.
- Definition of “large” fraction:
  - Origin AS: 80%
  - Transit AS: 20%

Using BGP RTG

- Ex Output: long ASCII records…

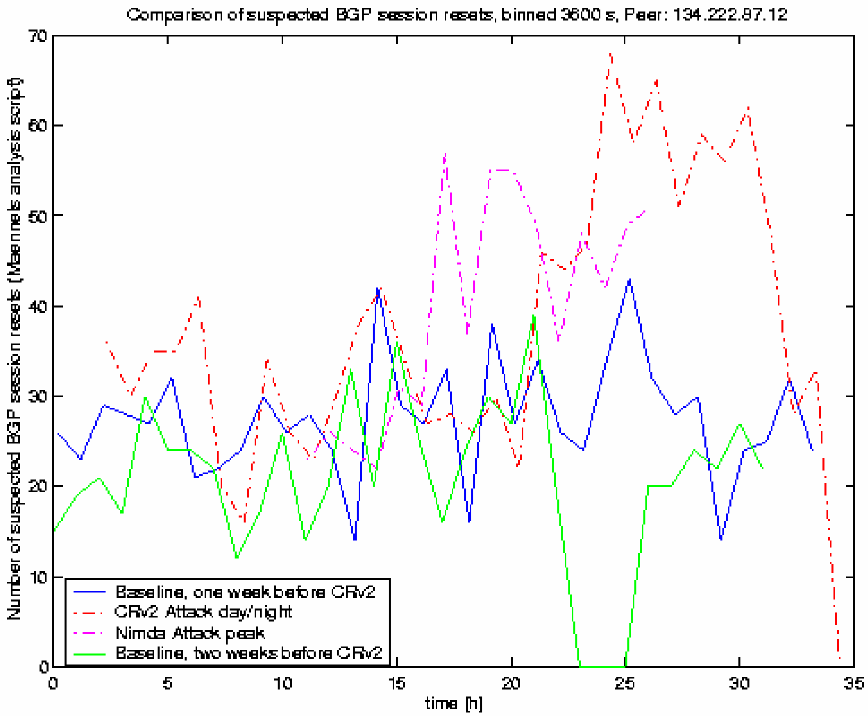
```
995487192|A|134.222.87.12|286|     12.32.72.0/23|286    209    17142
|IGP      |134.222.87.12||||NAG|             |32409.3303 |2  |:|4.  |27    |AA-DIFF|ASPath-way-shorter                         |209  |13904-
>17142|17142|only origin   |   286__89%_    209__86%_  13904__79%_  17142__75%_
| 2  |111   |0. |2.  |#3  |flapping     |100% | (17142)_*100%*both_     13904_**28%*oldAS      286____0%_both_      209____0%_both_
|2|13904 17142                    |    |   0.5|     instable    |  |       141|  4.|   1.|        56||0%|(5x 1x )
```

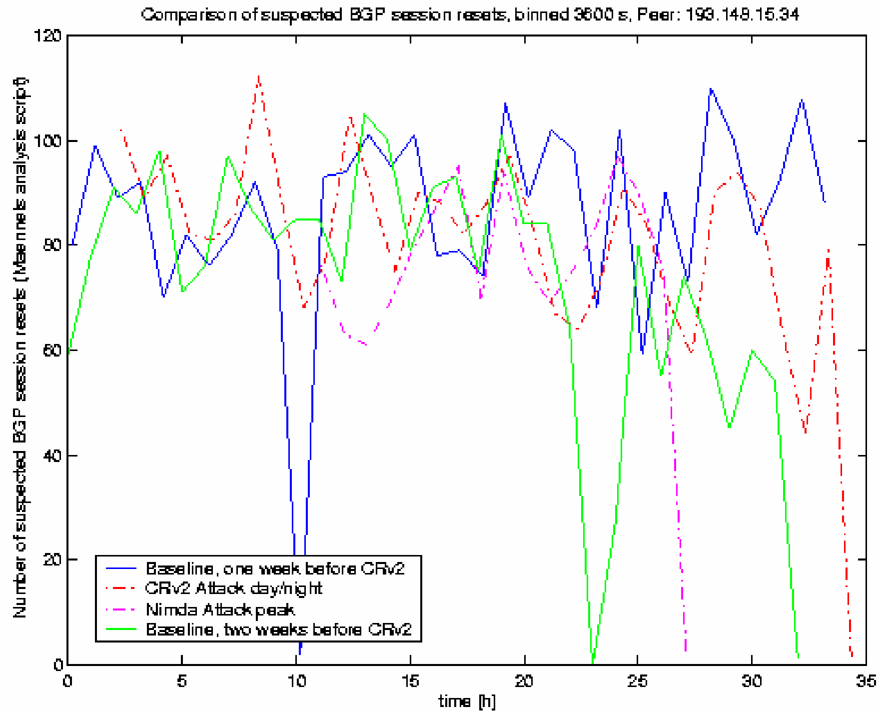
- Marks ASes "involved in suspected session resets"
  - Meaning "ASes having router(s) with session reset(s)"
- Appears to implicate too many ASes…
  - if transit AS, also appears to implicate originating ASes further down the path
  - Multiple markings of the same AS over different prefix update

⇒ We count the implicated ASes and check to avoid counting the same AS multiple times

Resets During Worms?

- Two example Peers





Observations

- One or two Peers appear to show an increase in “suspected resets” during the worm events compared to baselines
- However, the majority of data show no significant difference
- If the “globally observable” hypothesis is true, then we would expect a larger impact than we saw.

Conclusions

Some possible explanations:

- Inappropriate use of tool.
- Post-processing (counting) too restrictive.
- Bugs in the analysis code
  - who, us, write buggy code?
- "Unusual level of resets" hypothesis is wrong. (Possible, but not conclusively shown.)

$\Rightarrow$ Reliably detecting "remote" session resets seems difficult…

Some Comments on Heuristic

- "Small" ASes advertising only one or two prefixes will tend to be indicated whenever there's a change
- Updates could be due to internal route changes, not only resets
  - Not exactly clear how the BGP RTG tool deals with this

$\Rightarrow$ Could be under-counting due to update suppression from high transit connectivity

BGP-worm correlation: Just an artifact?

- Critique (Wang et al.): BGP-worm correlation was largely due to the table dumps induced by collection point session resets.
- Response: Such resets will certainly inflate the update counts. Let's filter them out and find out if there's still a correlation.
  - Wang et al. use a 25 minute filter

Filtering Table Dumps

- Hypothesis 1: Prefixes in a table dump are sent in monotonically increasing order.
  - If true, after an OPEN is seen, simply filter out all prefixes until a decrease is seen in consecutive prefixes.
  - It is false. For the RIPE peers, the prefixes are roughly in increasing order, but many are not.

Filtering Table Dumps

- Hypothesis 2: There are no repeated prefixes in updates until the full table dump is complete.
  - If true, after an OPEN is seen, simply filter out all prefixes until a repeated prefix is seen.
  - It is false. For the RIPE peers, some repeats are clearly seen during the middle of what is obviously a table dump.
    - It is not known if this is a bug or a new update mixed into the middle of the dump.

Filtering Table Dumps

- Hypothesis 3: A table dump should not invoke the rate limiting (MRAI) timer, therefore there should not be any significant gaps in time between advertisements in a table dump.
  - If true, after an OPEN is seen, simply filter out all prefixes until a gap on the order of the timer delay is seen.
  - It appears to be true. The number of prefixes counted between an open and a gap in time closely matches the previous table size heard from each peer.

“No-Gap” Filtering

- removed 2.4 million advertisements on Sept 18 (35.9%)
  - Wang et al. heuristic removed 2.7M (40.2%)

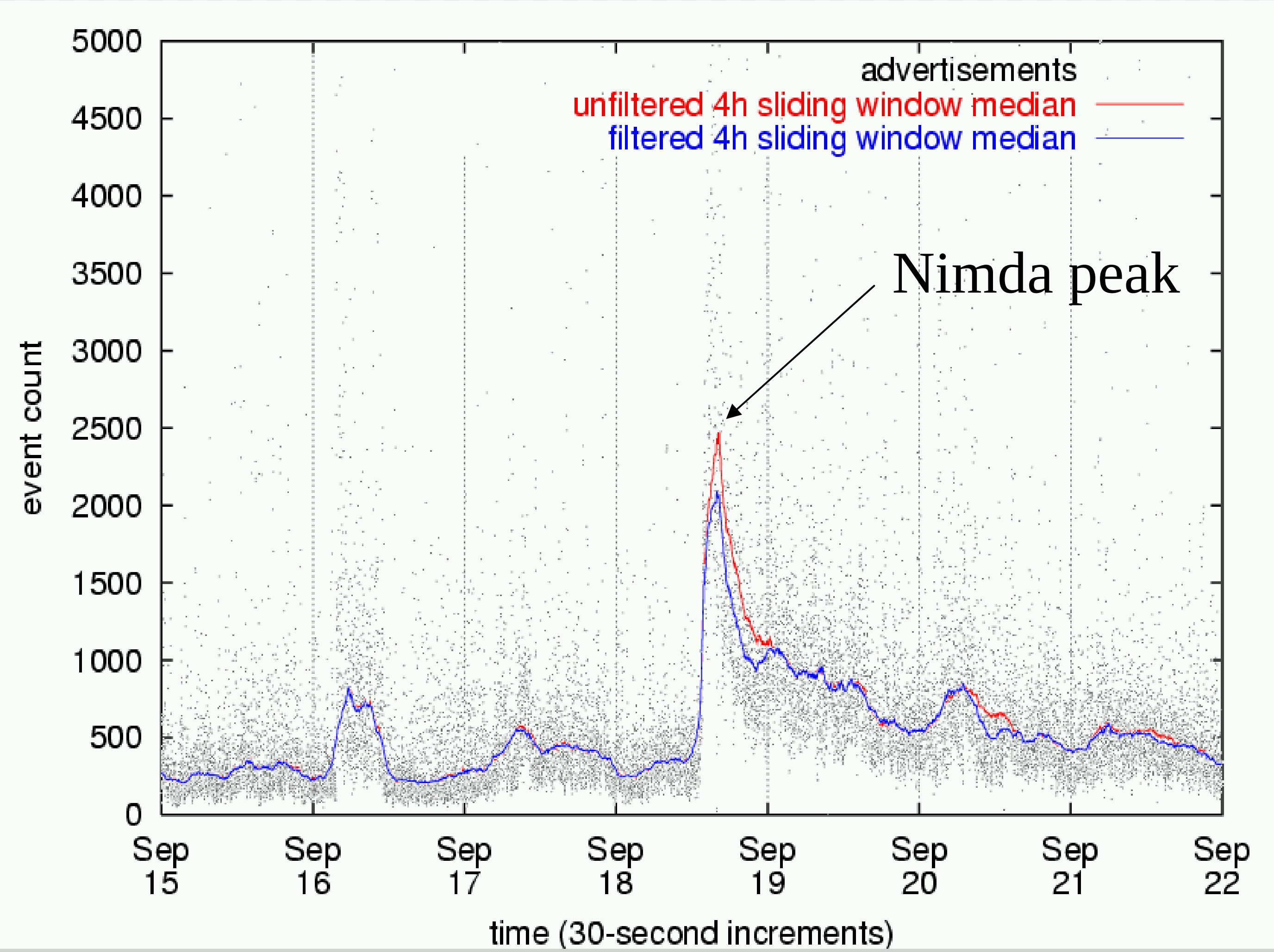
⇒ No OPENs on July 19 (Code Red)!

⇒ September 18 (Nimda):
4 hr sliding window median of prefix counts, before and after filtering is only slightly reduced

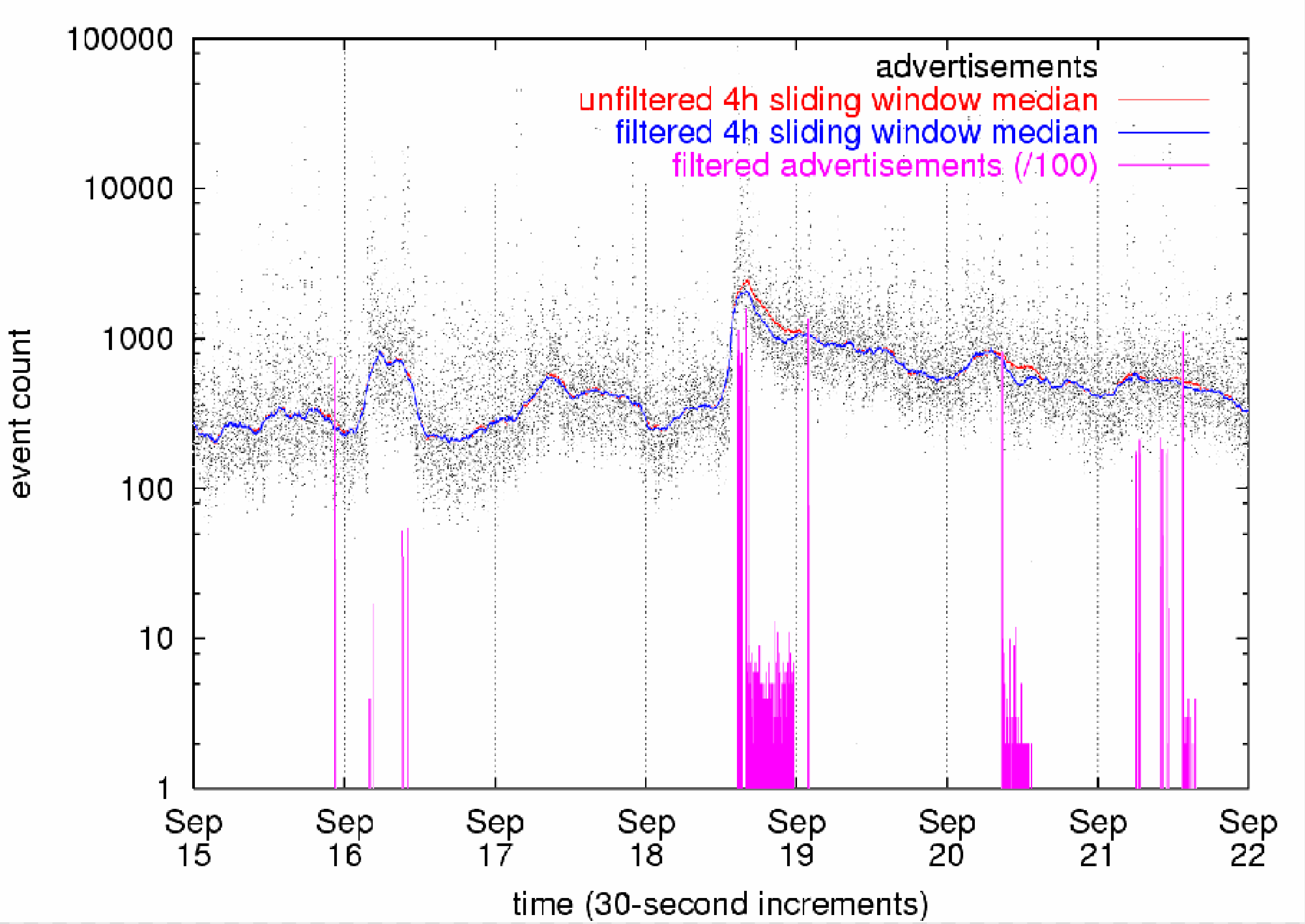
⇒ after filtering, there is still a strong correlation between the worm and total prefix advertisement counts

(September plot on next slide)

Before and After Filtering



Filtered Prefix Advertisements



Reset Detection

- We know that a reset results in updates, but how can we associate a subset of updates with a particular reset?
- Observe: A reset is composed of two distinct events:
  - session loss
    - typically results in a (possibly long) burst of advertisements; may end in either withdrawals or advertisements
  - session reestablishment
    - typically results in a burst of advertisements, possibly with some intermingled withdrawals, but always ends in advertisements

Hypotheses

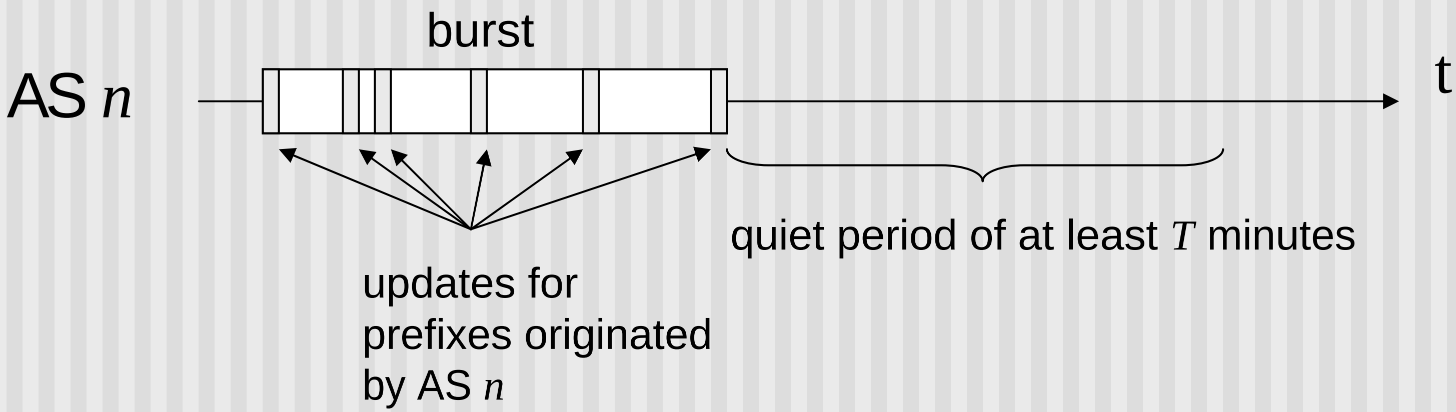
- session reestablishment will result in a burst of advertisements with common AS path prefixes
  - the final AS number in the prefix is the AS in which the reset occurred
- identifying resets is easier the closer the reset is to the collection point
  - less time for session to reestablish before new updates are propagated
  - more chance that the session was on the path used by the collection point

Ongoing Work

Using *per-AS update bursts*

- Motivation
  - Determining the root cause of single updates (from a single vantage point) is *very difficult*
    [T. Griffin, “What is the sound of one route flapping?”]
  - We try to circumvent these problems by
    - Coarser view: study *update bursts* rather than individual updates
    - Plan to correlate data from multiple viewpoints
      (Bursts, being coarser, seem more amenable to identification/correlation between viewpoints)
  - Also, resets/router crashes imply
    - Want to know when a whole AS is affected
      (unreachable/”detour” route) as opposed to single prefixes

Definition



- Burst of updates (advertisements or withdrawals) of prefixes originated by AS *n*
- Burst type:
  - advertise – if last seen prefix updates are all advertisements
  - withdraw – if last seen prefix updates are all withdrawals
  - undefined – otherwise (some prefixes advertised, some withdrawn)

  Meant to reflect “stable state” of AS after burst

Visualization

Driving questions

- Is there a qualitative difference in updates during worm events?
- Is it attributable to edge or core ASes?

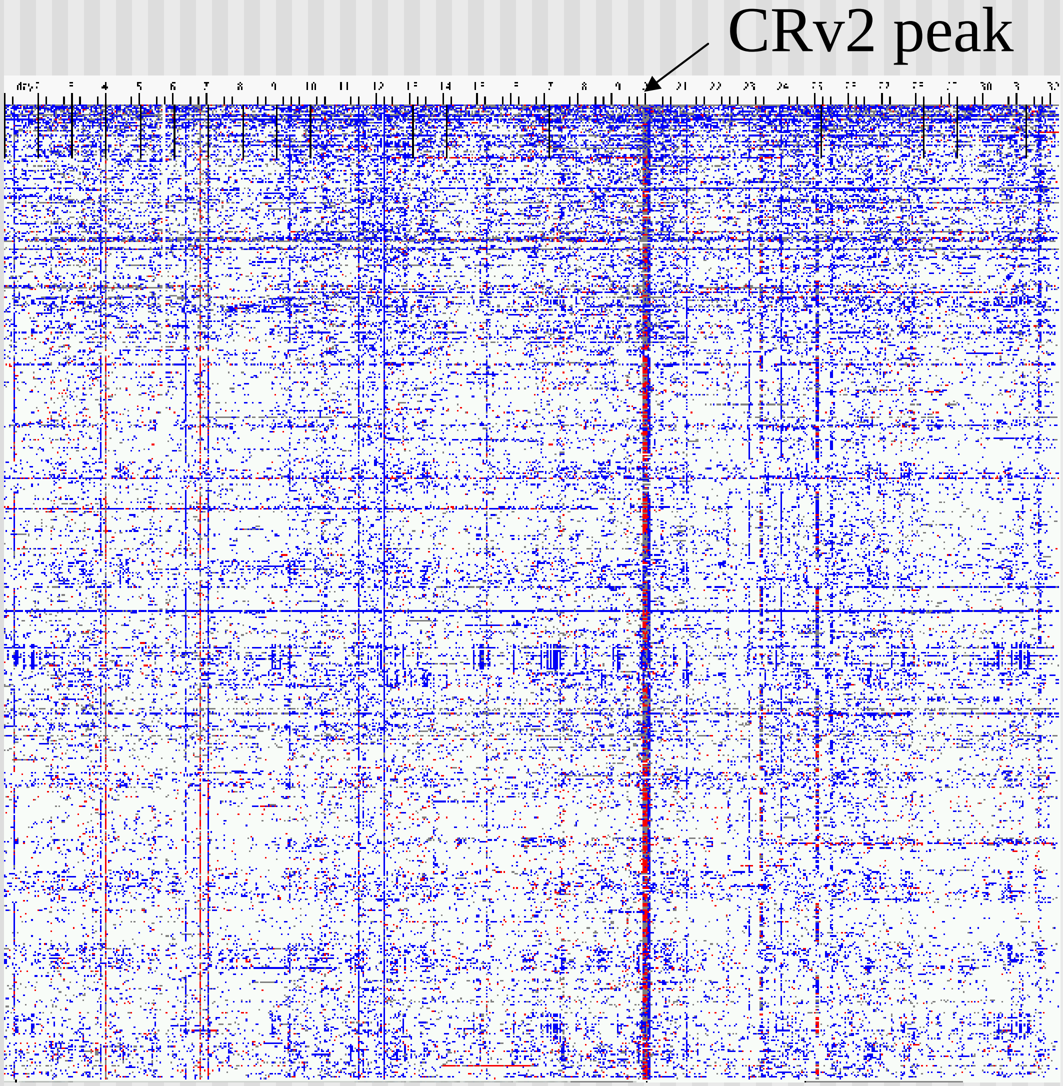
Why visualize?

- Try to provide a fathomable view as close to “raw data” as possible
  - Applying aggregate measures or statistics too early can be misleading…
    (discouraged by failed attempts to come up with statistics…)

⇒ Look at the collected bursts over single/multiple peers and for as many affected ASes as possible.

Data shown here is *after peer OPEN filtering*.

July 2001 – Code Red v2



- Peer 193.148.15.85
- X-axis: time [days]
- Y-axis: one line / AS
  - Sorted by outdegree, and ordered:
    - core ASes towards top
    - edge ASes towards bottom
- *T* = 20 mins
- Color key:
  - White – quiet
  - Blue – advertisement burst
  - Red – withdrawal burst
  - Gray – undefined burst type

Some Observations

Differs from other graphs/studies in that it

- breaks data down per originating AS – attempting to show “state”
- attempts to show differences between “core” ASes and “edge” ASes

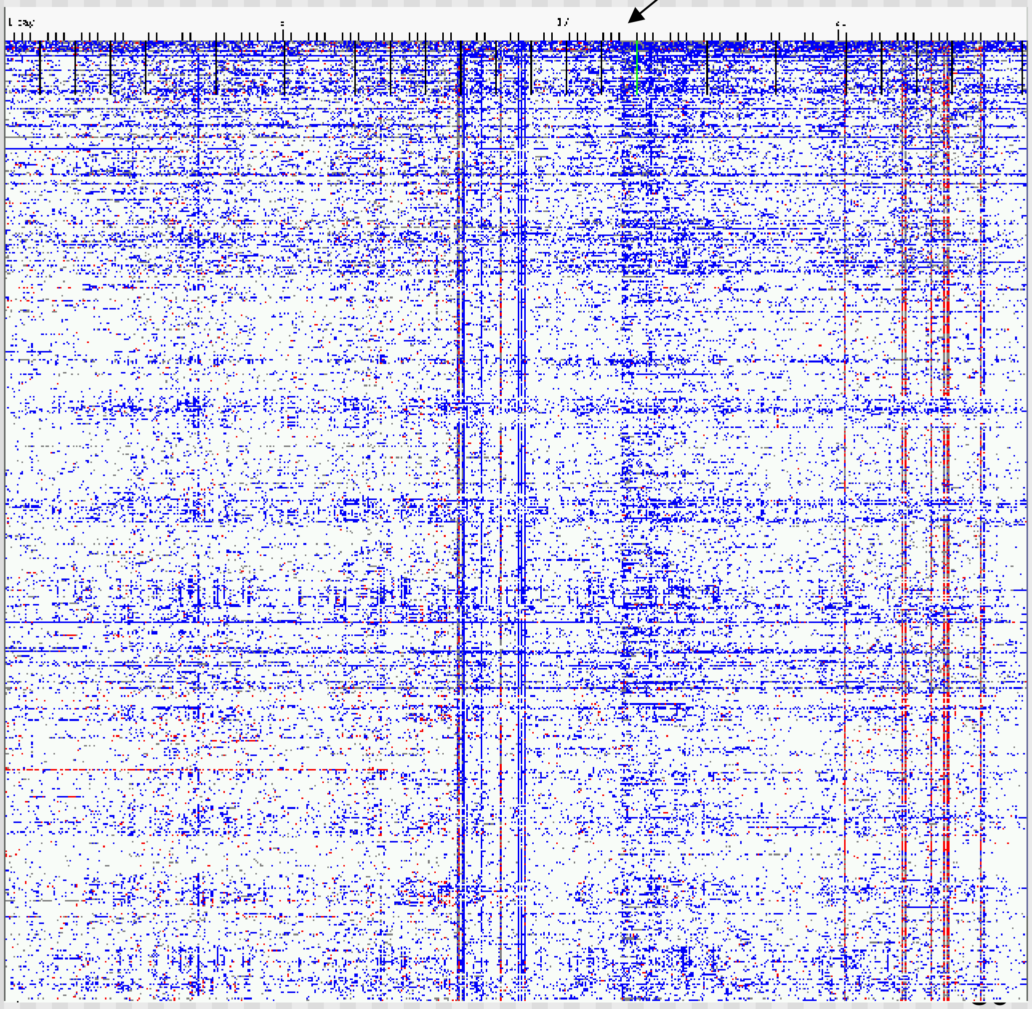
After peer OPEN filtering: (actually no OPENs on the 19th)

- Unusual event at this peer on evening of 19th, correlated with the CRv2 worm.
  - Very dense updates affecting many (most?) ASes
  - More extended in time than most other similar events – which appear likely to be session resets in ASes that are not immediate collection point peers
- Other peers show similar indications, although less distinctive.
  - Thus, visible over all peers – “global”

Sept 2001 - Nimda

- Same peer: 193.148.15.85
- Appears different from updates during Code Red v2 event:
  - No similar distinct withdrawals
  - Prolonged “wave” (several days) of advertisements – similar timescale difference as the worm events

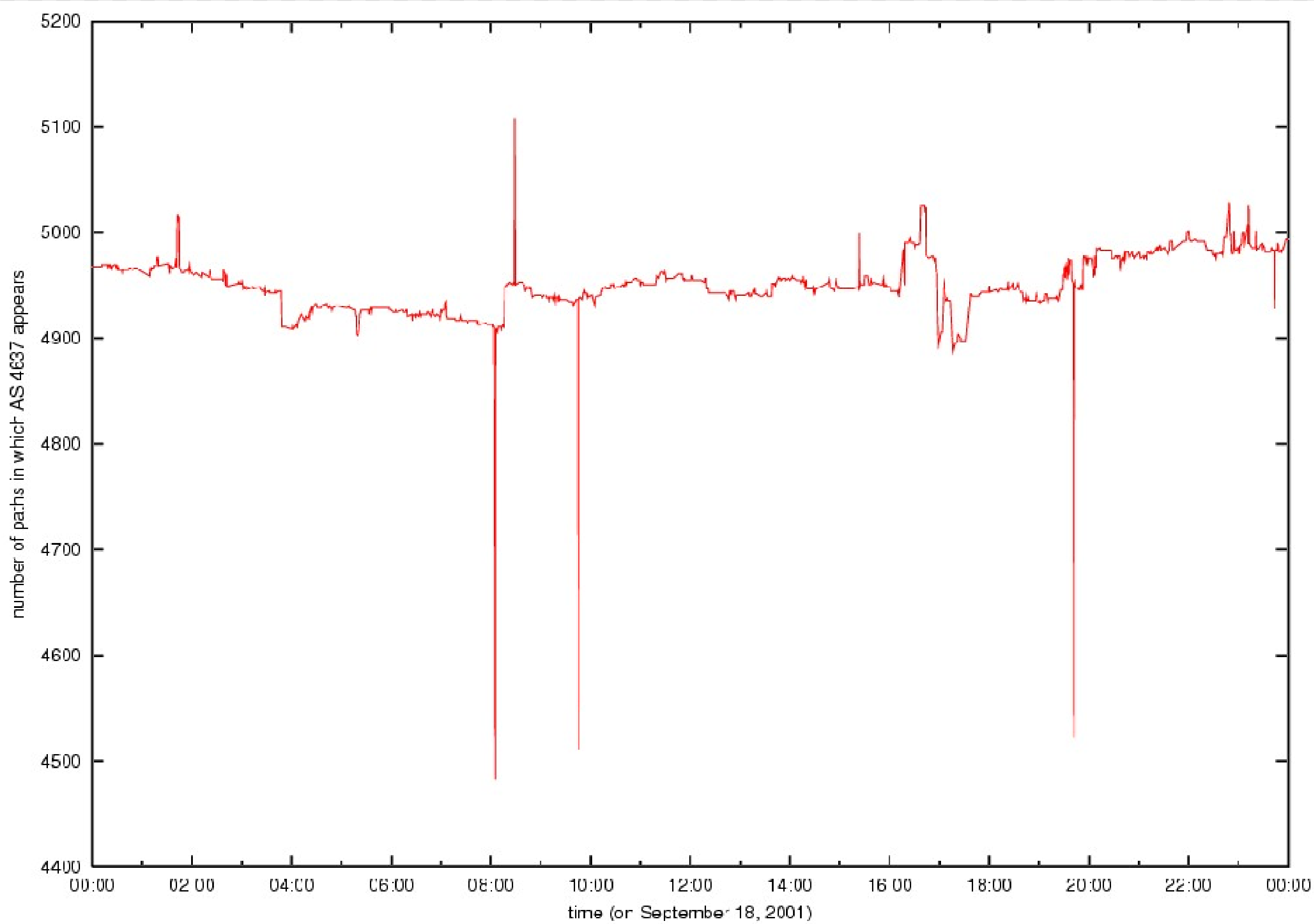
Nimda peak



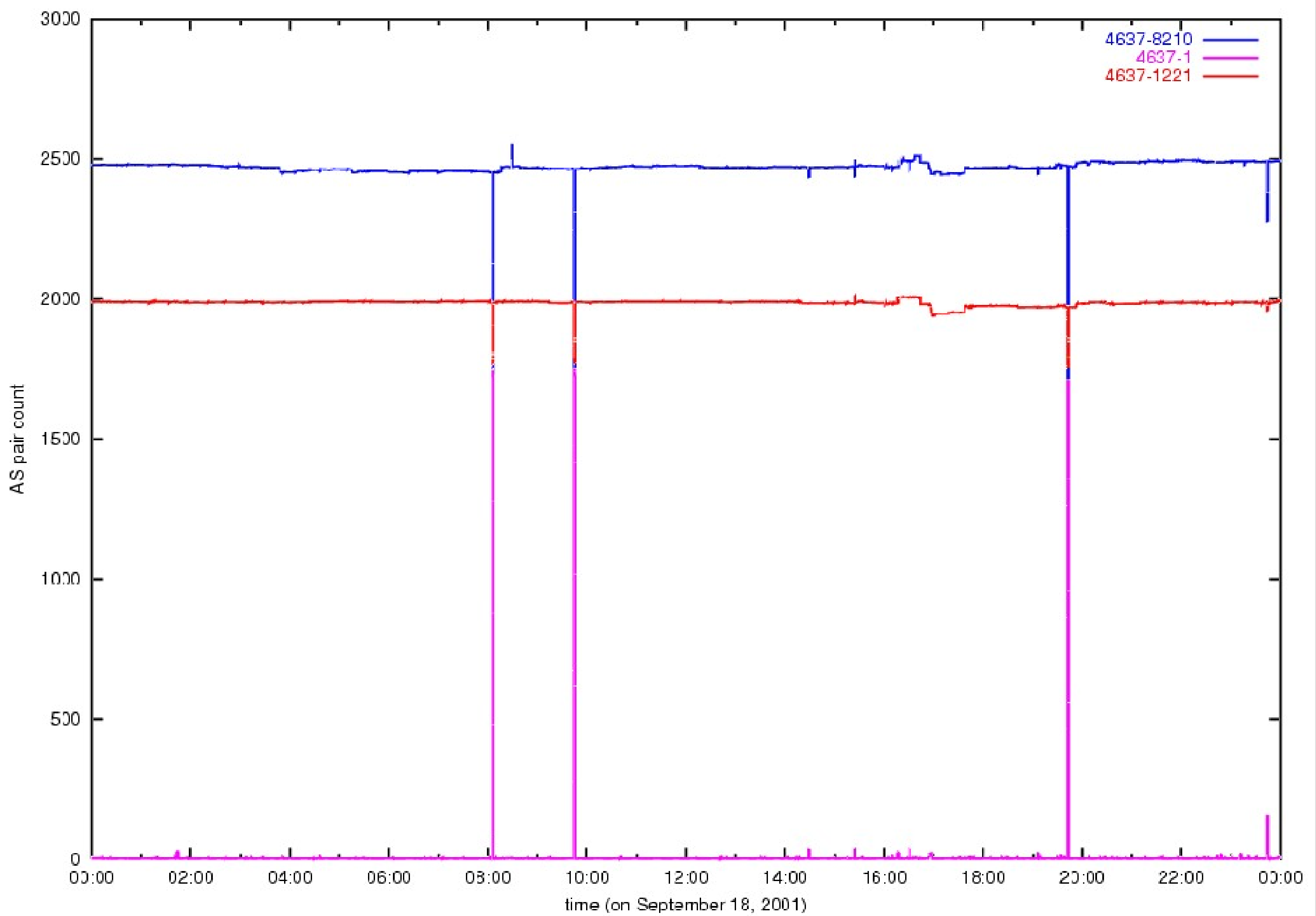
Next Steps:Hiccup Detection

- How to pinpoint instability creators? Look for AS pairs in flux
  - For each AS look for high variance in number of paths containing it
  - Example : 4637 during nimda attack

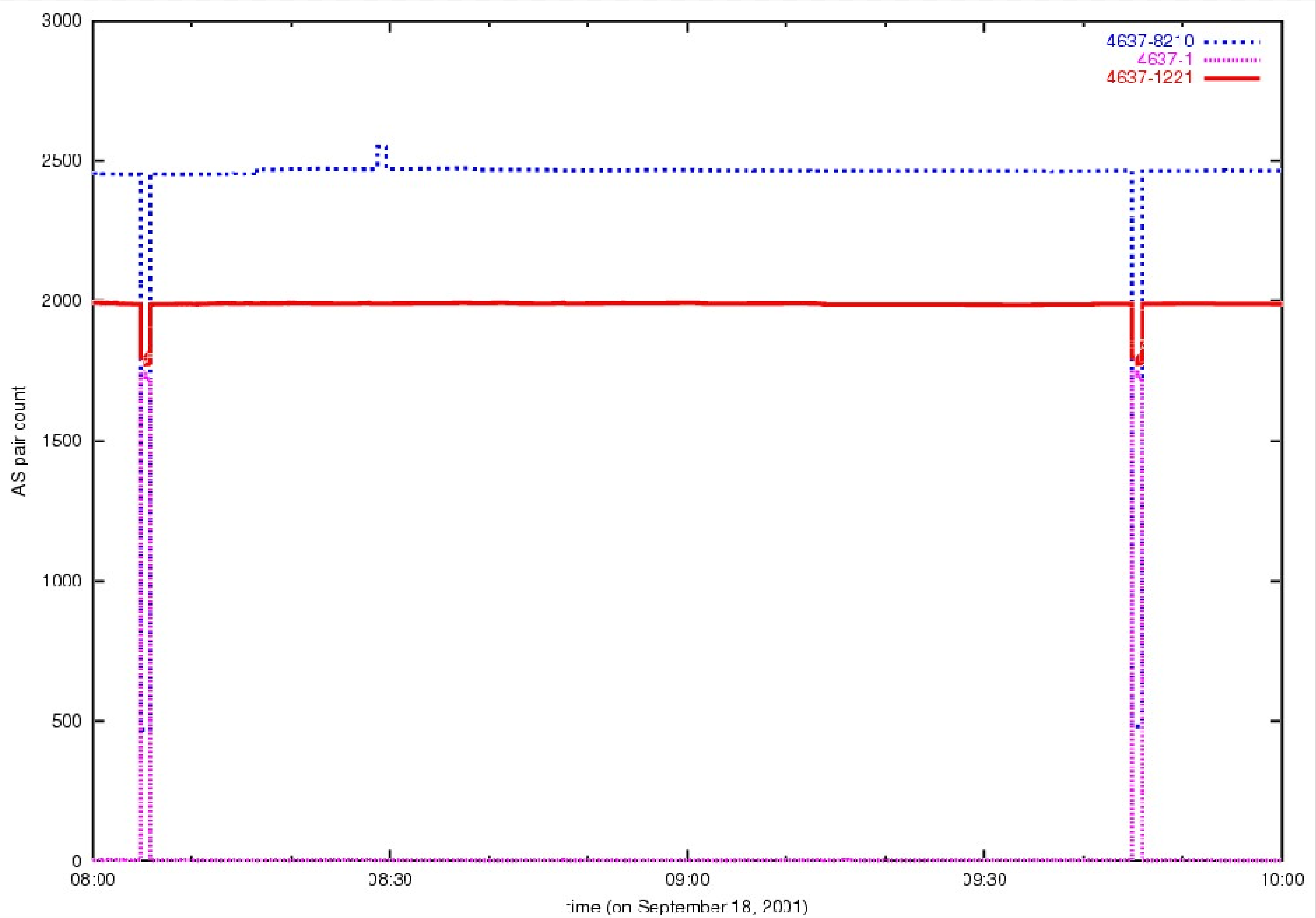
All paths containing 4637



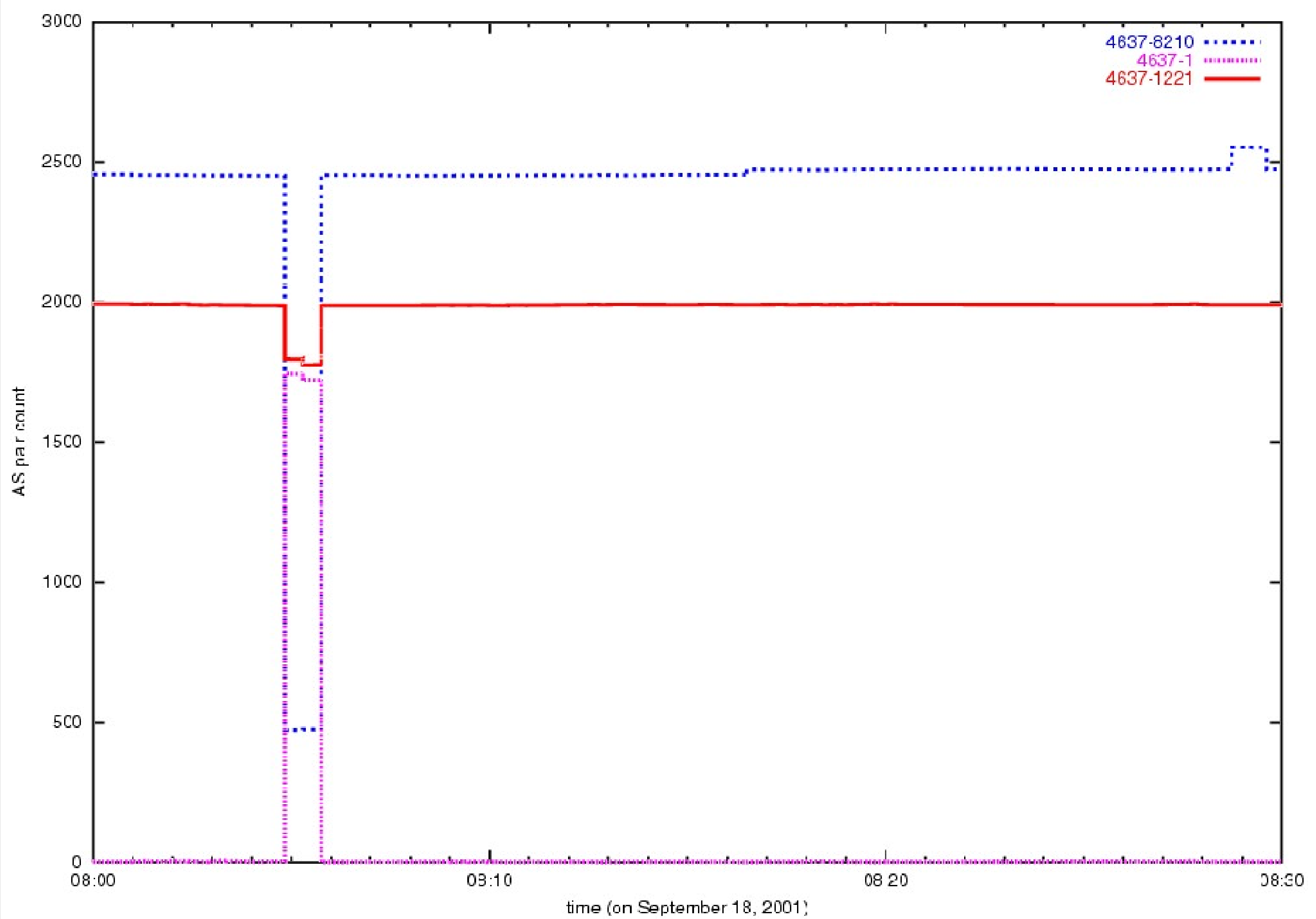
Individual paths containing 4637



Microscope


4637-8210
4637-1
4637-1221
3000
2500
2000
1500
1000
500
0
AS pair count
08:00
08:30
09:00
09:30
10:00
time (on September 18, 2001)

Electron Microscope



Routing under attack

The worm surges were accidents. What could happen if someone *attacked* routers?

- Wang et al. suggest that most of the surge is explainable by instability in a <u>*few*</u> edge ISPs
- What if someone went after BGP with malice in their heart?
  - All it takes is high utilization at high priority

Summary

- Have developed epidemic models "Part 1" (www.cs.dartmouth.edu/~nicol/papers/mascots2002.pdf, or www.cs.dartmouth.edu/~nicol/papers/mascots2002.ps.gz)
- Collection point peer OPEN filtering
  - Validated heuristic – (results similar to [Wang et al.])
  - Does not change conclusions of an advertisement surge during worms
- Locating distant BGP instability creators (including session resets) is not easy…
  - Explicitly trying to avoid some of the problems indicated by [Griffin] through:
    - Looking at coarser structure: bursts rather than single updates
    - Correlating multiple vantage points (planned)