D-CART: Dynamic Change Analysis with Routing Traces

Changements de routes

Pascal Mérindol, Pierre David, Jean-Jacques Pansiot : **Université de Strasbourg**
François Clad, Pierre François : **Cisco Systems**
Stefano Vissicchio : **Université catholique de Louvain-la-Neuve**

merindol@unistra.fr
http://icube-reseaux.unistra.fr/dcart

D-CART

The practical problem

Design of the platform

Measurements & Analysis

Theoretical formulation of the solution

Problem definition

Constraint c associated with a loop L

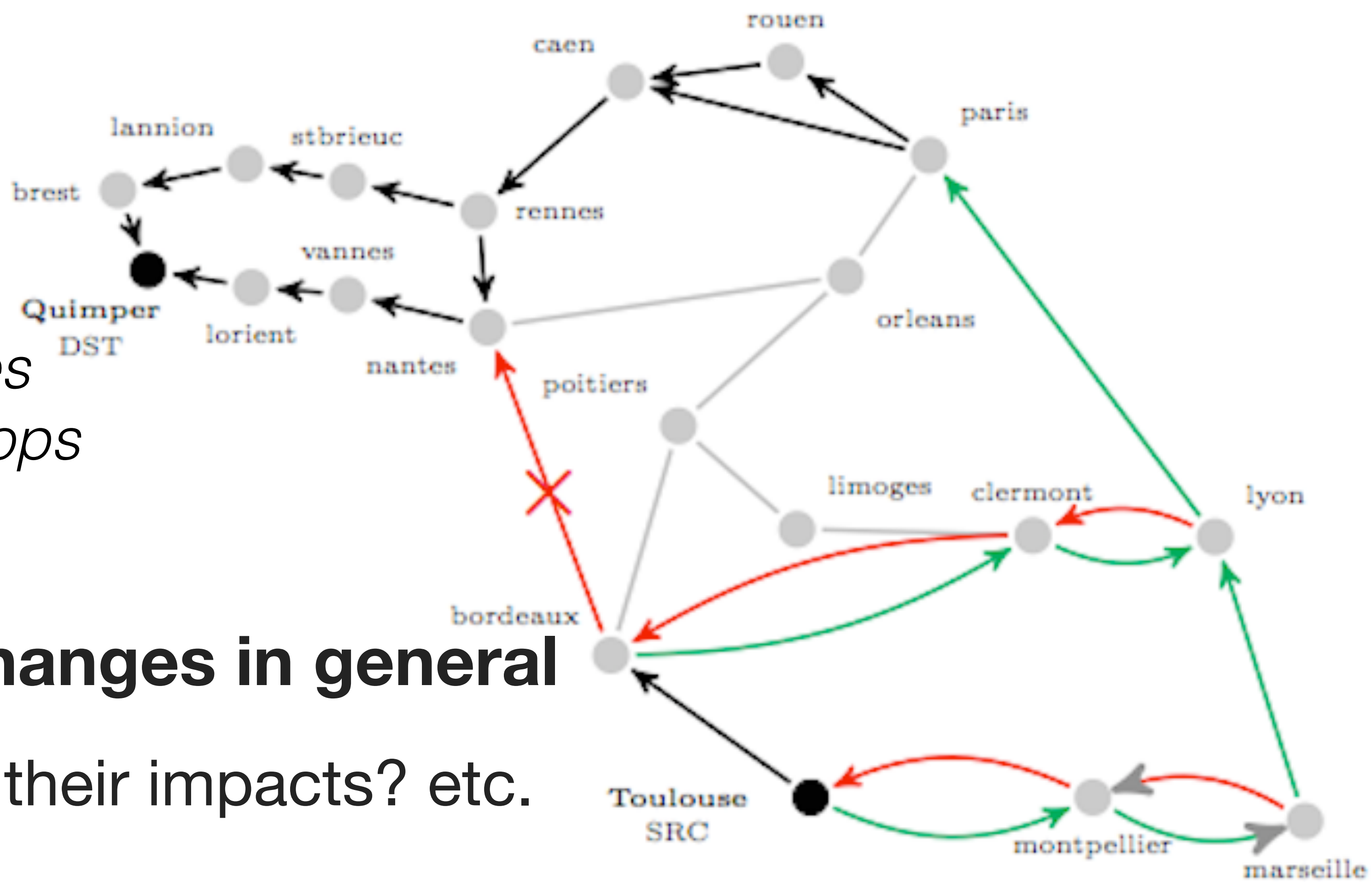
$$c := \left(\min_{\forall x \in L}(\Delta(x)), \max_{\forall x \in L}(\Delta(x))\right)$$

Greedy **B**ackward **A**lgorithm

A Brief History of our Collaboration with RENATER

- **2011-14: Reveal, study and solve transient routing loops in LS-routing**
  - Ph.D. Thesis topic of *François Clad* (Unistra -> Post Doc Cisco)
    - Theoretic and incremental solutions: no protocol changes required!

A loopy illustration on RENATER:
the link Bordeaux-Nantes fails and the combination between pre- and post-routes triggers up to four transient forwarding loops for the pair Toulouse -> Quimper!



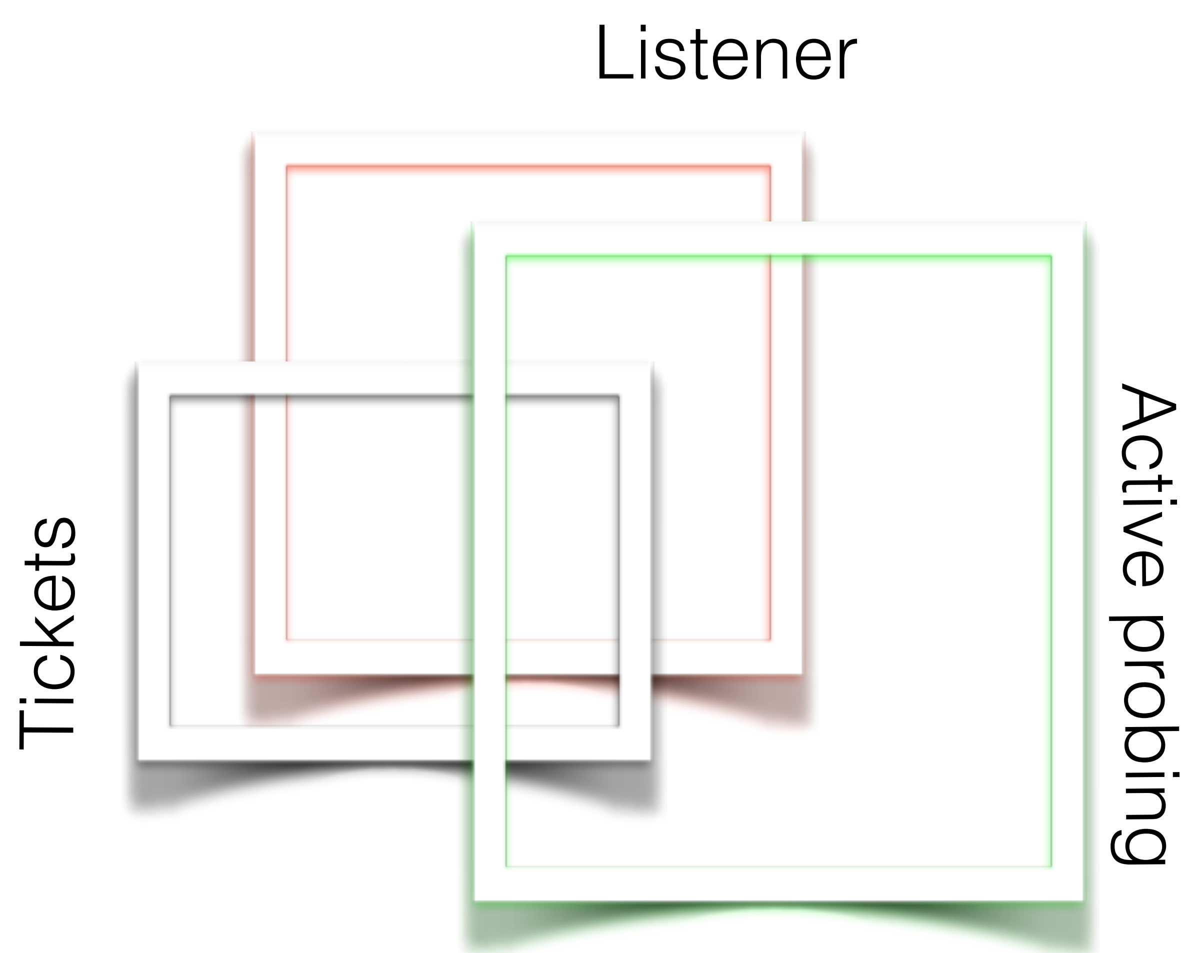
- **2014-16: Understand routing changes in general**
  - What are typical loss durations? their impacts? etc.
    - How minimize such periods?

Why D-CART?..and why this way?

- **Troubleshoot your IP network**
  - IS-IS configuration in particular & possible extensions for OSPF, MPLS, BGP, etc
- **Improve performances of your IP network**
  - modify configurations for improving the reachability and routing delays
    - avoid routing loops and reduce cut/blackhole periods due to routing changes
    - use better routing paths according to the traffic load (weight changes)
- **Develop specific monitoring primitives and re-use existing tools**
  - open software provided by the networking research community (GPL)!
- **Avoid measurement interferences**
  - dedicated but common hardware infrastructure at a (really) limited cost

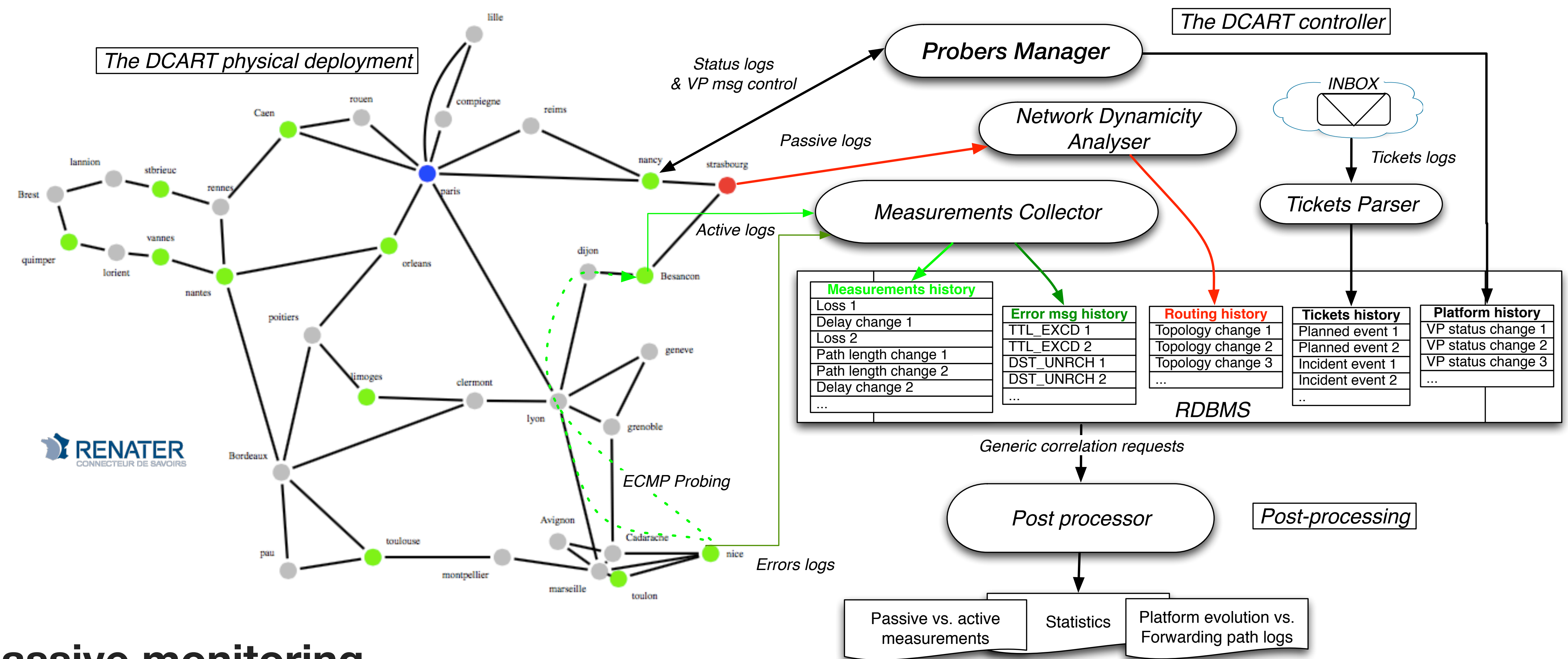
D-CART: main characteristics

- **3 Sources of data that can be more or less correlated**
  - IS-IS routing states of all routers
  - Active directed ping-like measurements
    - + error messages
  - Maintenance and incident tickets



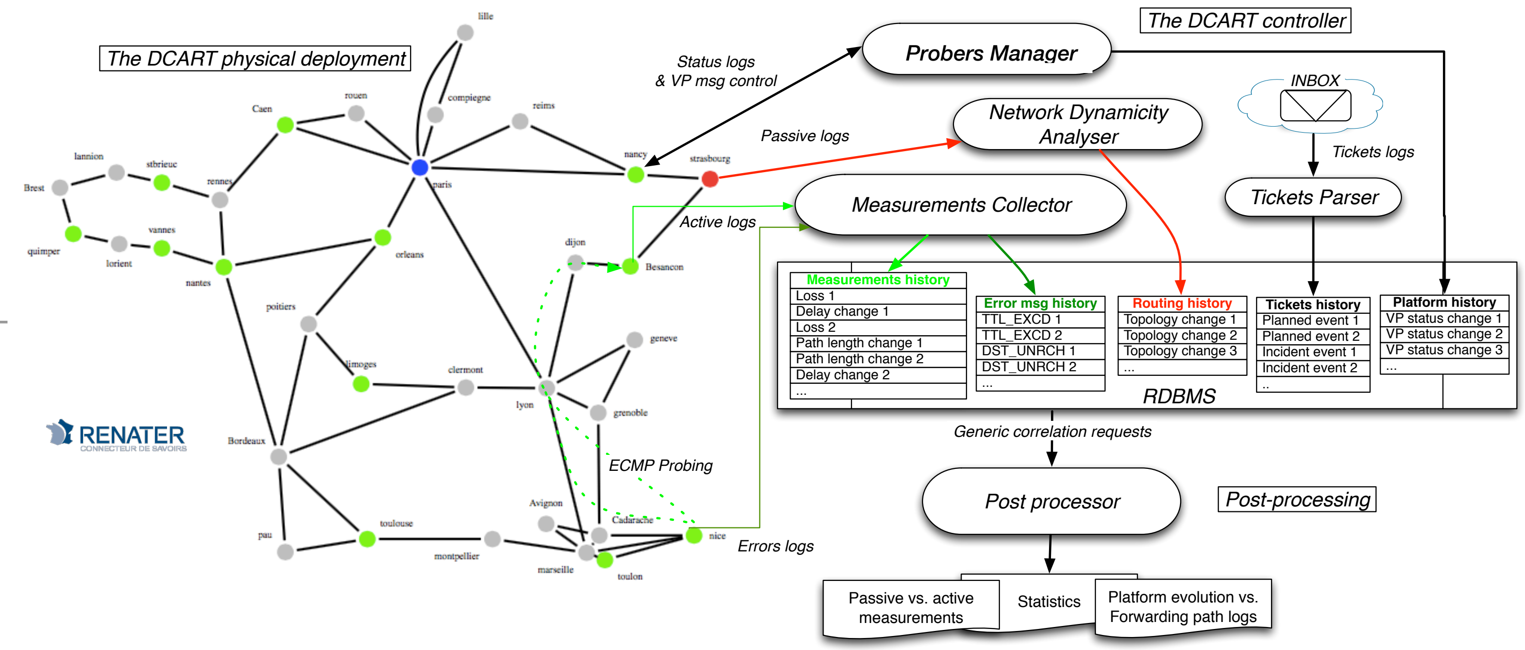
- **Open Platform Design**
  - using low cost Raspberry-PI hardware directly plugged to routers
  - targeting IPv4 intra-domain routing events in particular…
  - …but extensible in any directions in theory!

How D-CART works?.. The big picture!



- **Passive monitoring**
  - a silent IS-IS listener (an extension of Sprint's work) and a tickets feed
- **Active monitoring**
  - a bunch of 16 Ras-PIs logically organized as a full-mesh (4 are located in PARIS)

How D-CART works?



- **Main Software Components**
  - <u>Probers Manager</u>
    - control/check probers and record their status
  - <u>Network Dynamicity Collector</u>
    - manage the listener output: filter and associate LSPs
  - <u>Measurements Collector</u>
    - manage most interesting routing events: *error messages* (TTL Excd. and Dest. Unreach.), *losses & de-sequencing*, *delay changes* and *path changes*.
  - <u>RDBMS</u> (PostgreSQL)
    - record events and ease correlations between them
  - <u>Post processor</u>
    - perform statistics about events: join distinct sources of data

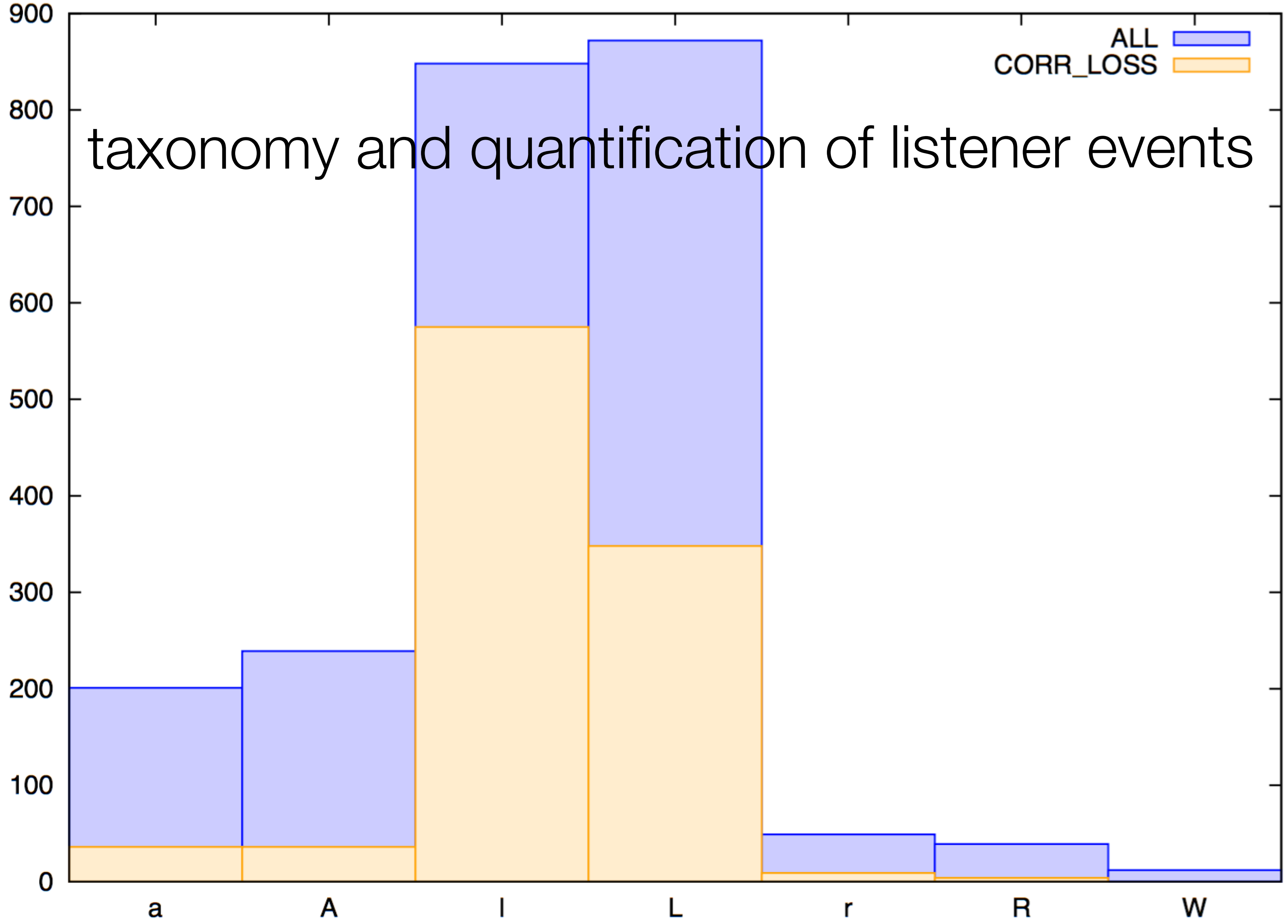
Our set of specific measurement primitives

- **D-CART current design**
  - smart directed ping-like probing
    - -> get evidences and locations of transient routing loops!
  - Equal Cost Multi-Path aware probing
    - -> measure accurately all possible paths
      - need to use multiple IP address (load balancing performed at the IP level)
  - NTP synchronisation (10ms at worst, ~1ms in practice)
    - -> to compute one way delays and allows correlations among data sources

- **D-CART current calibration**
    - probing frequency: the highest possible -> 40 ms…mainly a hardware tradeoff
    - towards a low amount of logs: scalability (30GB for 4 months of overall data)

Listener vs. Losses

- **What is the share of routing events triggering losses?**

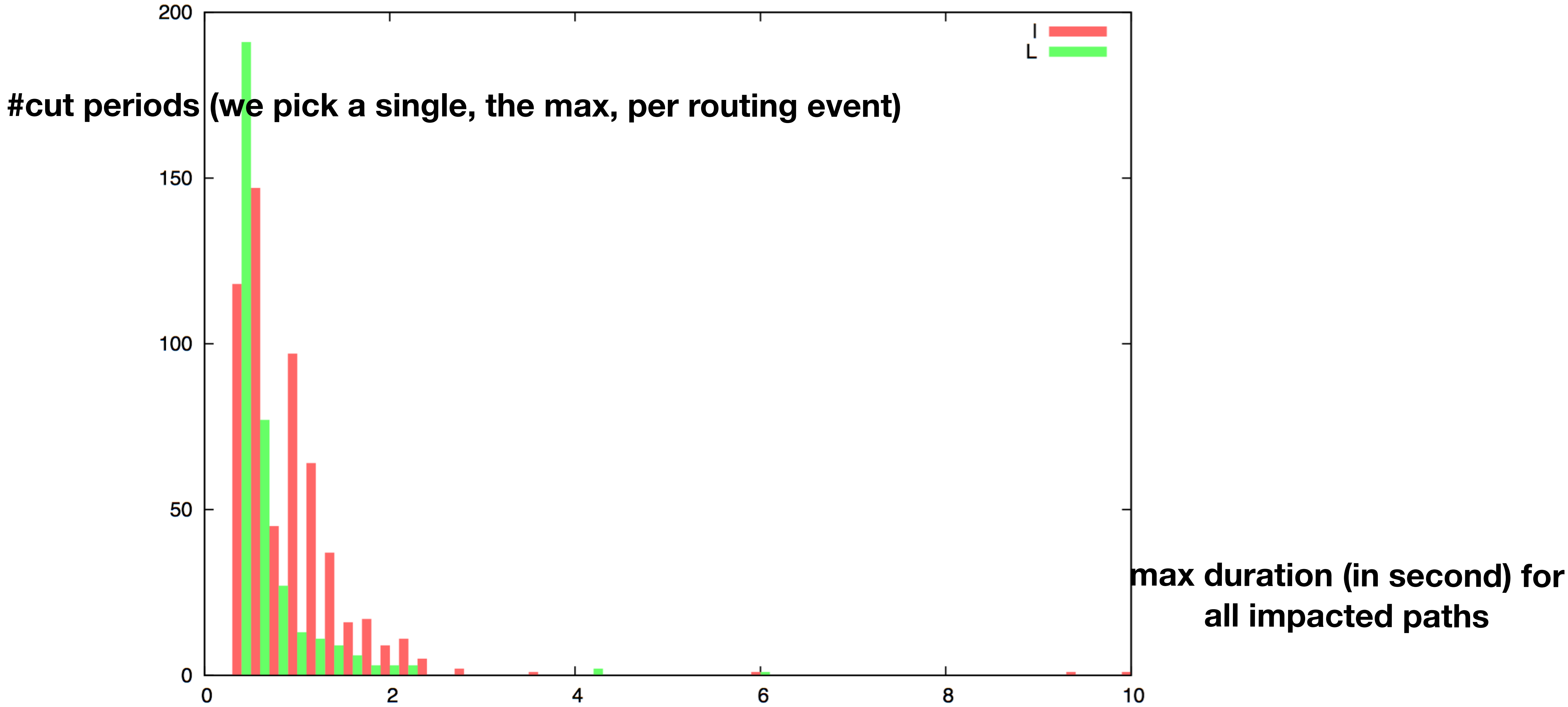


- *a: adjacency down*
- *A: adjacency up*
- *l: (bi-dir) link down*
- *L: (bi-dir) link up*
- *r: router down*
- *R: router up*
- *W: weight change*

- Even **L**ink up triggers many losses (and so micro-loops?)!
- why such a difference with **r/R**outer and **a/A**djacency changes?
  - for a/A, we can't observe all of them…(no probers for each leaf)

Listener vs. Losses (link focus)

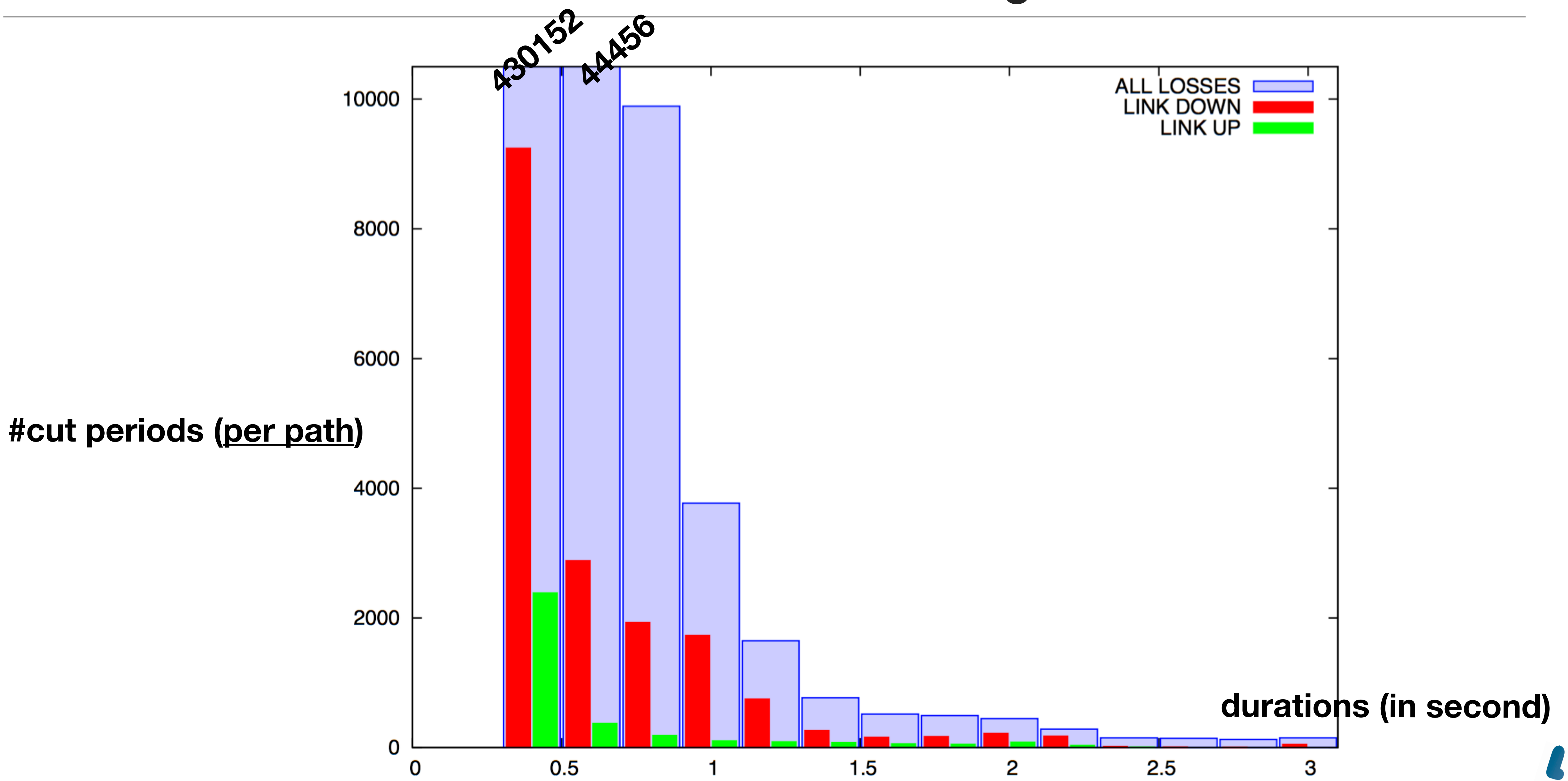
- **What is the distribution of worst losses (max durations) related to routing events?**



- For sure, it is worst for link down than for Link up…
  - mainly due to blackhole periods occurring for down events

Losses vs. Listener (link focus)

- **What are the distributions and shares of routing events related to losses?**

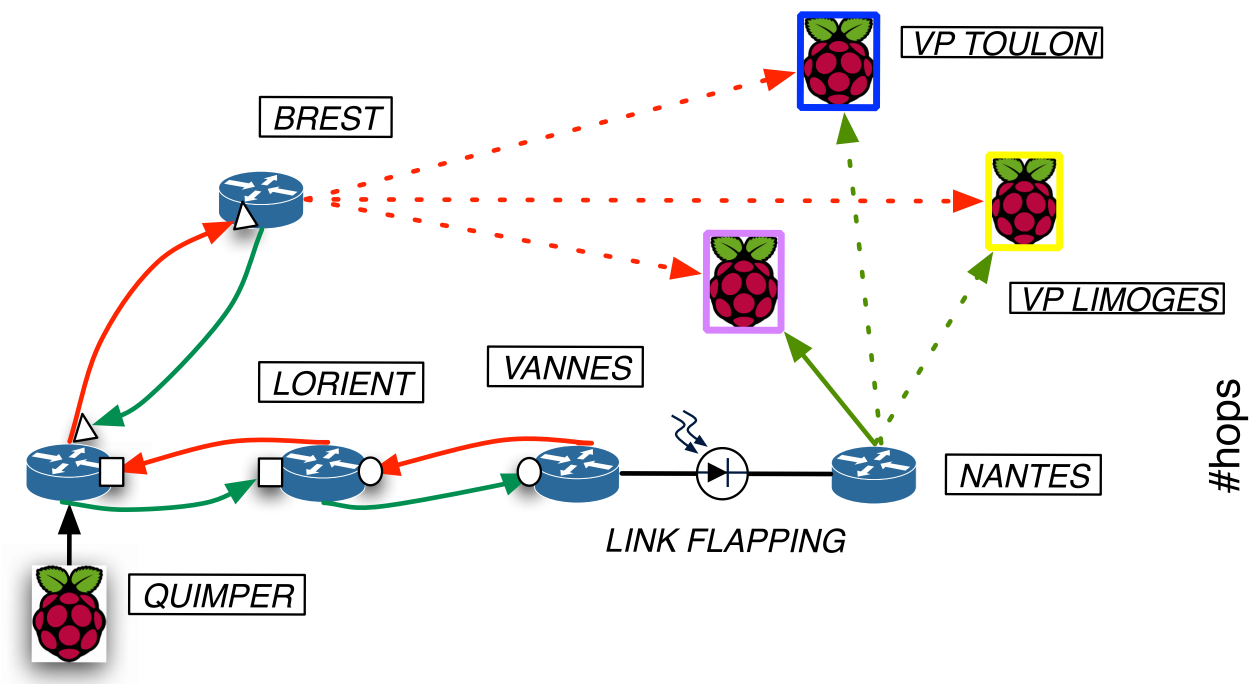


- The vast majority of short-time losses are due to congestions…
  - …long ones are “hopefully” mainly related to routing events!

An illustration of listener vs. probing correlation

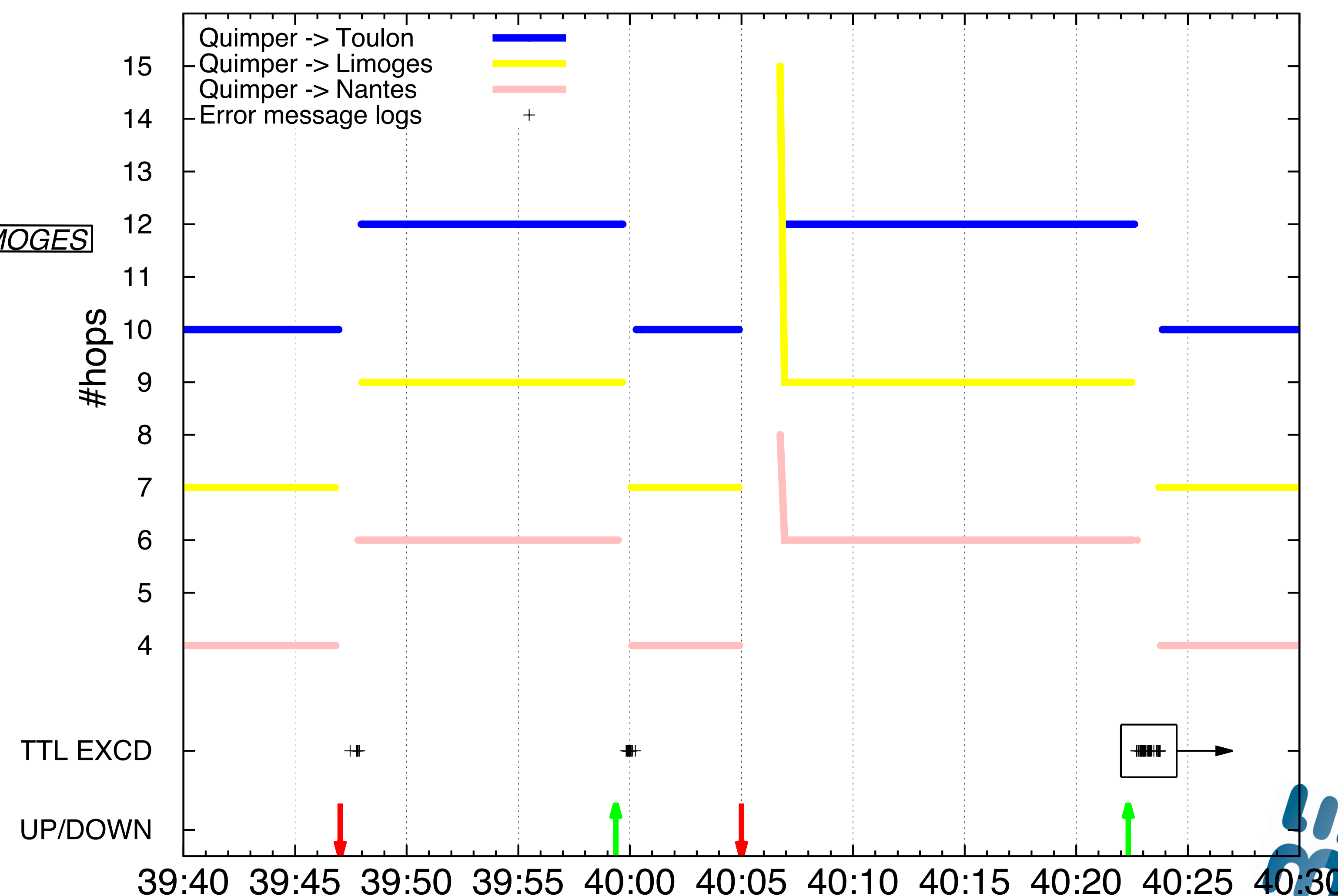
LOOPS INSIDE

- **About the origin of routing change related losses**



The "setup": a link is flapping between Nantes and Vannes -> 3 potential loops in a row

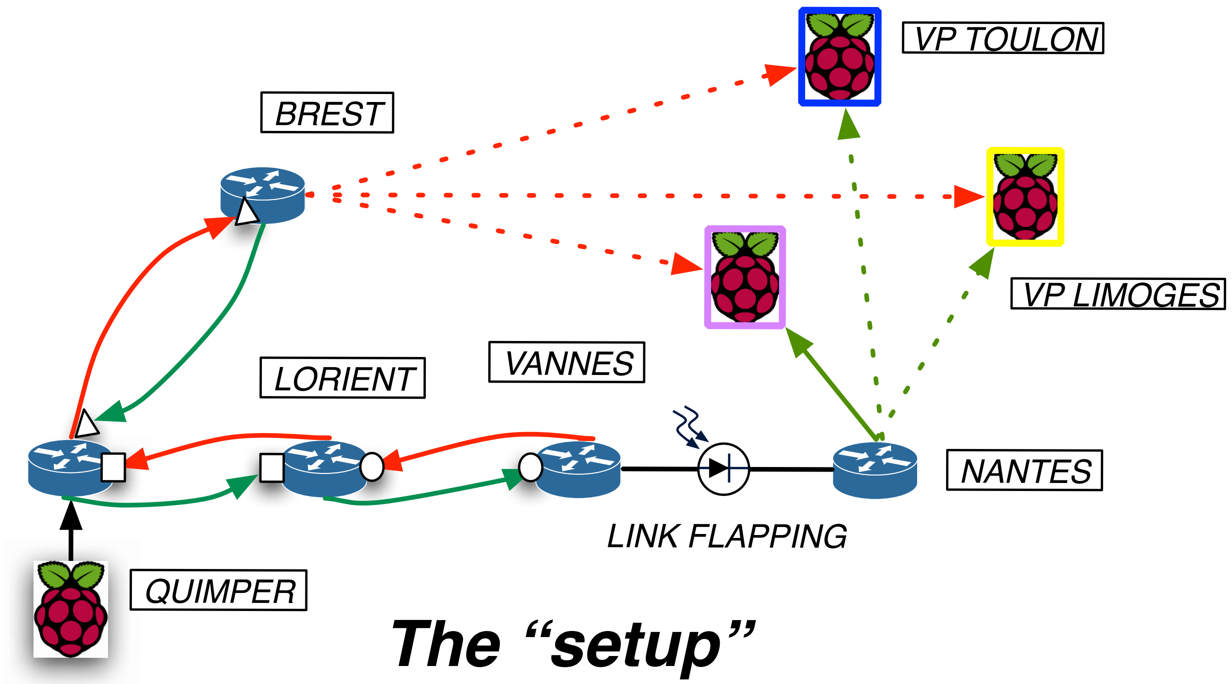
Path changes and side effects



The longest cut occurs at the last link UP : only due to micro-loops!

Zoom on the last link up

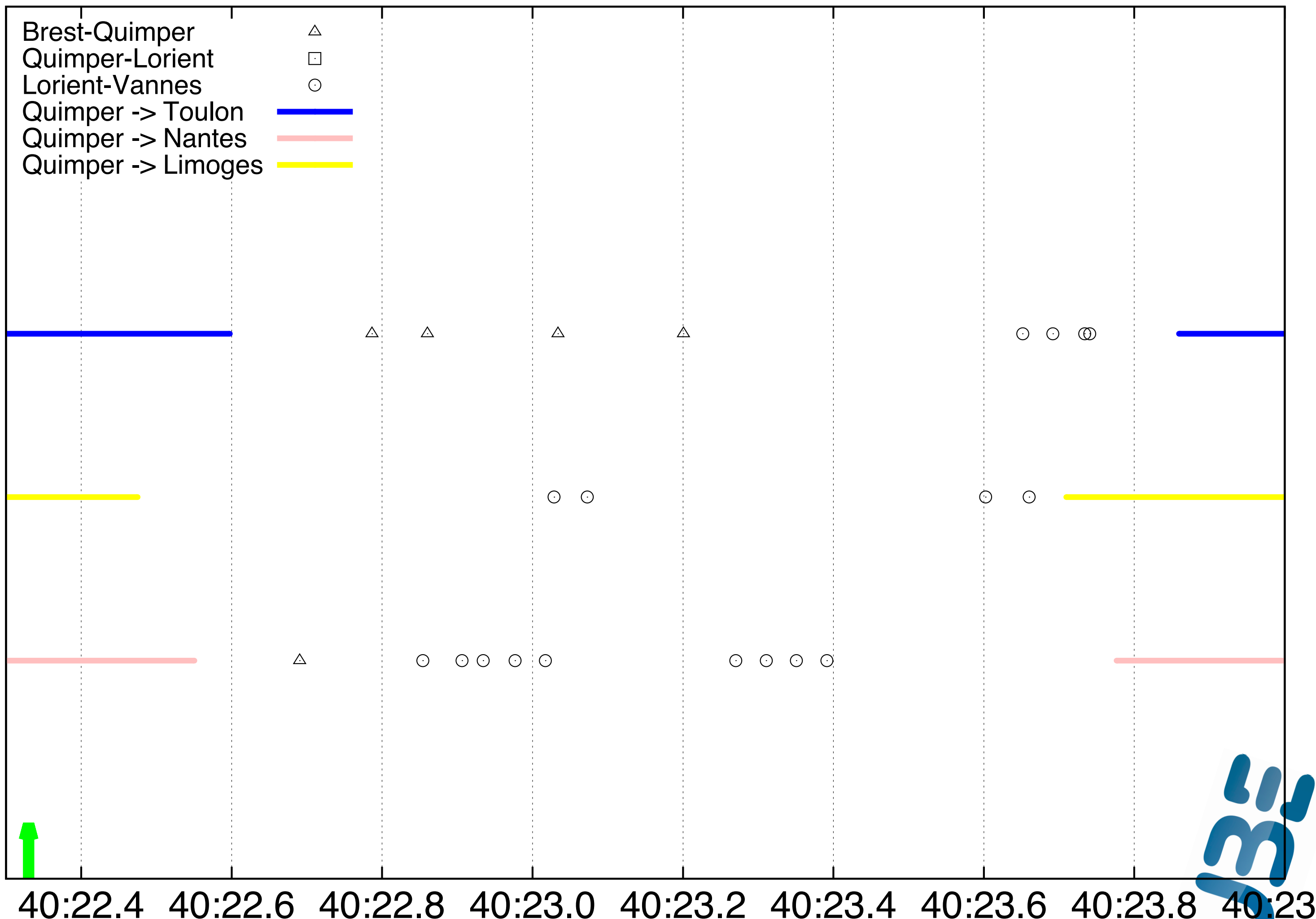
A BIG UP LOOP INSIDE



The "setup"

Timelines for the 3 destination prefixes (the probers) and loop locations analysis

Cut period: micro-loops zoom



More than one second of traffic interruption! <- FIB update order

Simple comparison between listener and tickets #events?

- **The listener provides a better granularity than tickets**
  - **and it is automatic!**

type	nevents	rel_freq	freq_per_day
A	239	0.20568	2.096
L	872	0.75043	7.649
R	39	0.03356	0.342
W	12	0.01033	0.105

2.36 link up/down per day

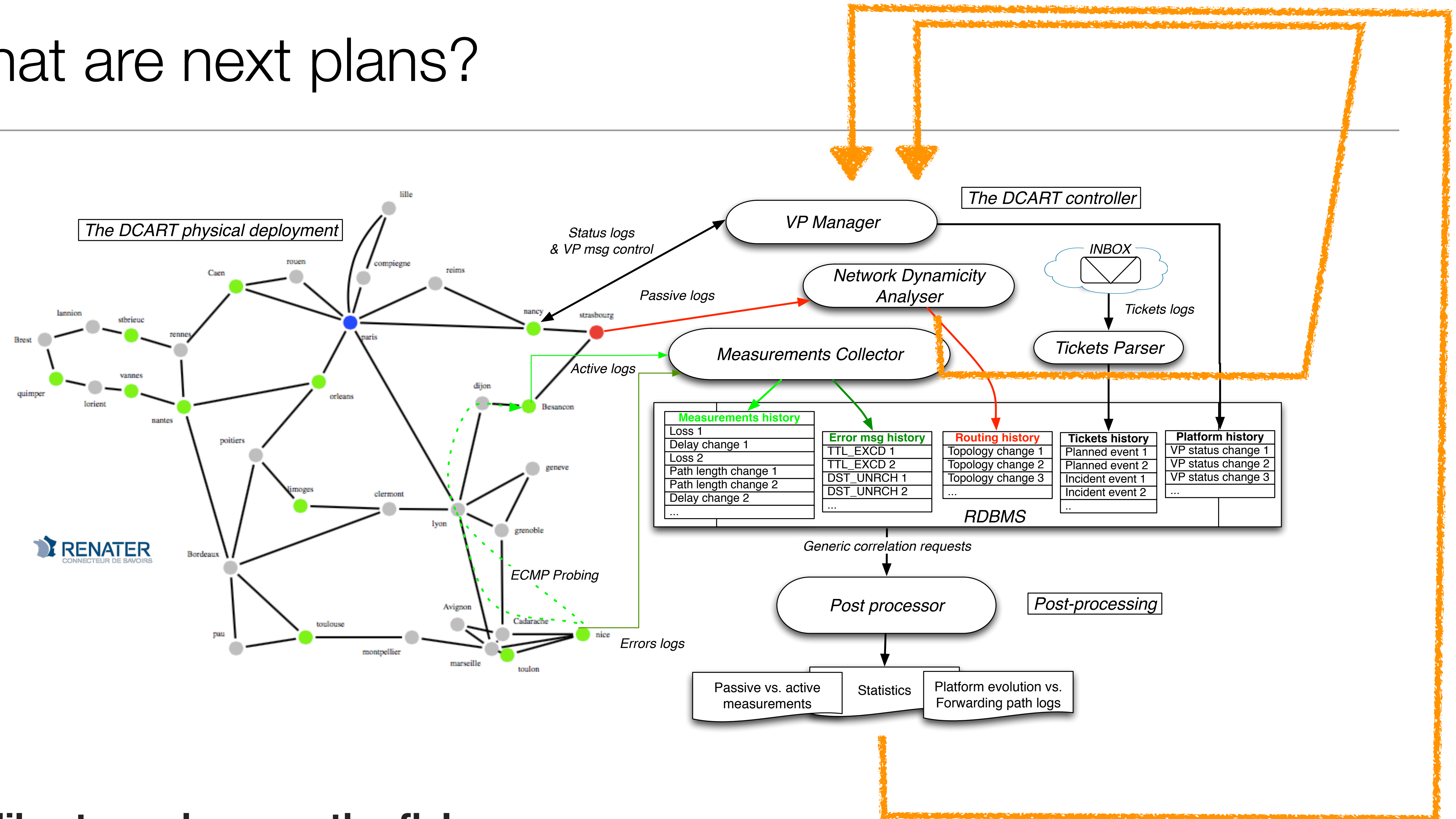
aggregation per link per day // flapping

Listener events

ntick	freq_per_day	type	subtype
52	0.29050	incident	link
12	0.06704	incident	router
72	0.40223	maintenance	link
33	0.18436	maintenance	router

Tickets events

What are next plans?



- **Recalibrate probers on the fly!**
  - e.g. modify probing frequency of a subset of probers according to the listener
  - e.g. reprogram probers with specific measurement mechanisms for planned events

About the D-CART platform: summary and future works

- **Basically, it is an open software monitoring platform**
  - the hardware doesn't matter that much…
    - …as long as it is not specific and powerful enough
  - it is generic enough to support all kind of specifics measurements
    - it ensures flexibility, scalability and extensibility
  - we get numerous loop evidences and even more!
    - automatic ticketing system, failure prediction (flapping), bugs detection, etc.
- **We envision to extend it in several directions**
  - across multiple IGP networks…Geant and more?
  - comparing IPv4 and IPv6 traffic forwarding performance differences
  - TCP-like measurements: routing changes side effects on real applications
  - focus on BGP modifications effects, etc.

D-CART

The practical problem

Theoretical formulation of the solution

Design of the platform

Measurements & Outcomes

Problem definition

Greedy **B**ackward **A**lgorithm

Constraint c associated with a loop L

$$c := (\min_{\forall x \in L}(\Delta(x)), \max_{\forall x \in L}(\Delta(x)))$$

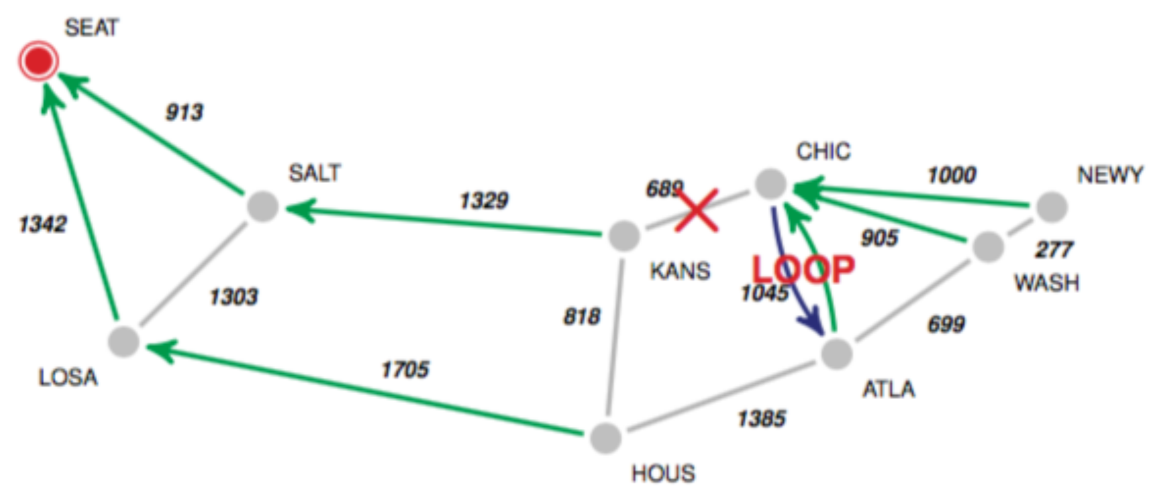
About our solution

- **Objective: get rid of transient forwarding loops**
  - dealing with all kind of routing changes: up/down/weight changes * link/router
  - at least for planned events (but works well in any cases in theory)

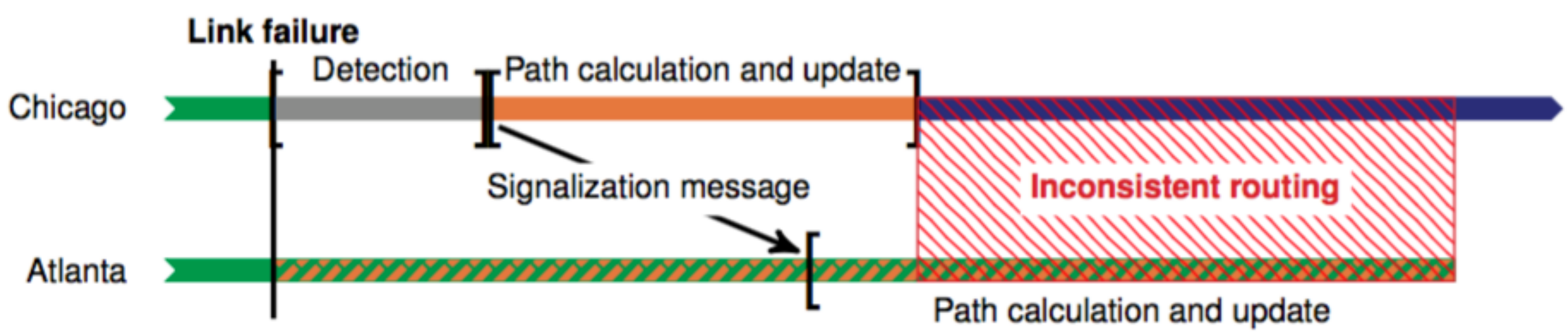
- **Constraint: design a practical solution**
  - No protocol changes
    - incremental solution
  - No explicit synchronisation among routers required
    - ≠ oFIB or other schemes
  - Efficient and scalable at all levels
    - ≠ ships in the night that works and is designed for the whole network

D-CART -> GBA

- **But first, let us recall the problem on a detailed but simple illustration…**



Shortest paths towards Seattle



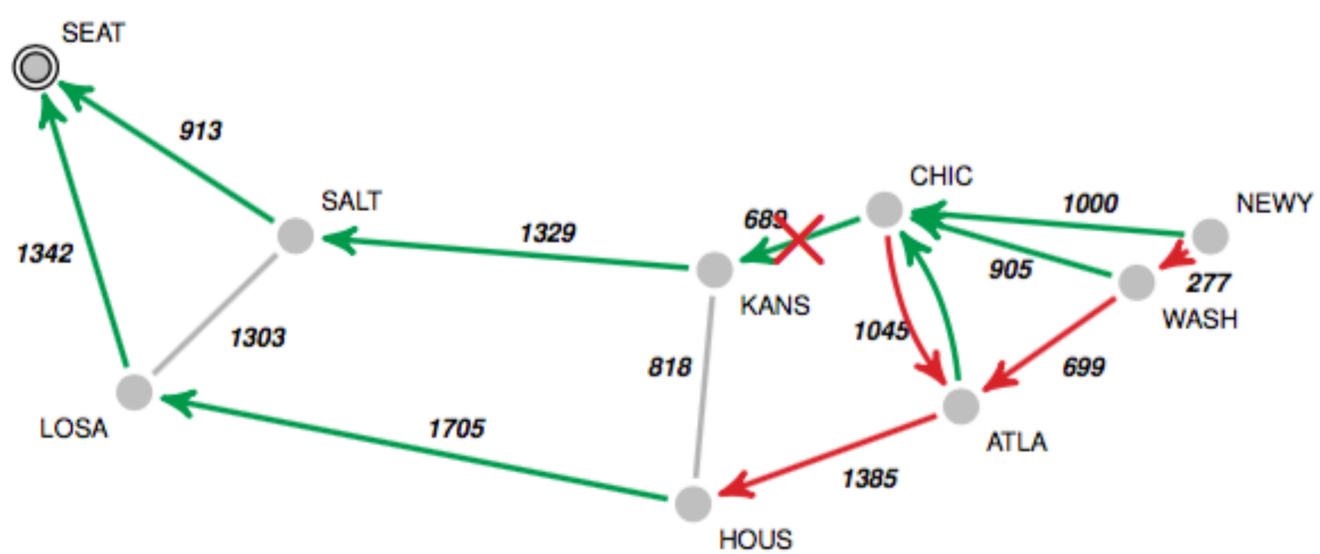
Convergence of the routers at Chicago and Atlanta

- **Can we play with link weights to progressively shift the traffic and update router's FIB in the right order?**

The “Delta”: equilibrium values at which routes change

For a given destination d, we define for each router a pivot increment, denoted $\Delta_d(x)$:

$$\forall x \in N, \ \ \Delta_d(x) = C'(x,d) - C(x,d)$$



x	$C(x)$	$C'(x)$	$\Delta_{SEAT}(x)$
SEAT	0	0	0
LOSA	1342	1342	0
SALT	913	913	0
HOUS	3047	3047	0
KANS	2242	2242	0
CHIC	2931	**5477**	2546
ATLA	3976	**4432**	456
WASH	3836	**6176**	2340
NEWY	3931	**6453**	2522

General definitions:

$G(N,E,w)$	Directed weighted graph representing the network
$C(x,d)$, $C(x)$	Cost of a shortest path (*distance*) from x to d before the change
$C'(x,d)$, $C'(x)$	Cost of a shortest path (*distance*) from x to d after the change

Problem definition (system constraints)

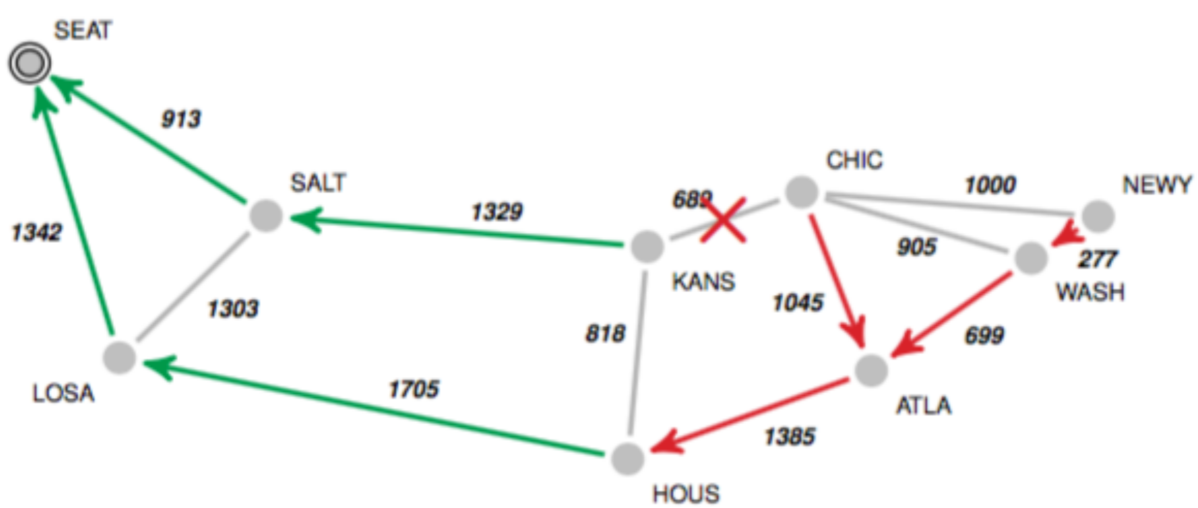
- **Notion of sequence of increments**

Theorem

A monotonic weight update sequence S prevents a transient loop $L = \{x_1, x_2, \ldots, x_1\}$ for a destination d, if and only if there exists $e \in S$ such that:

$$MIN_{\forall x \in L}(\Delta_d(x)) < e < MAX_{\forall x \in L}(\Delta_d(x))$$

The sequence must contain a weight update that makes one router involved in the loop to completely reroute, while another is still in its initial routing state.



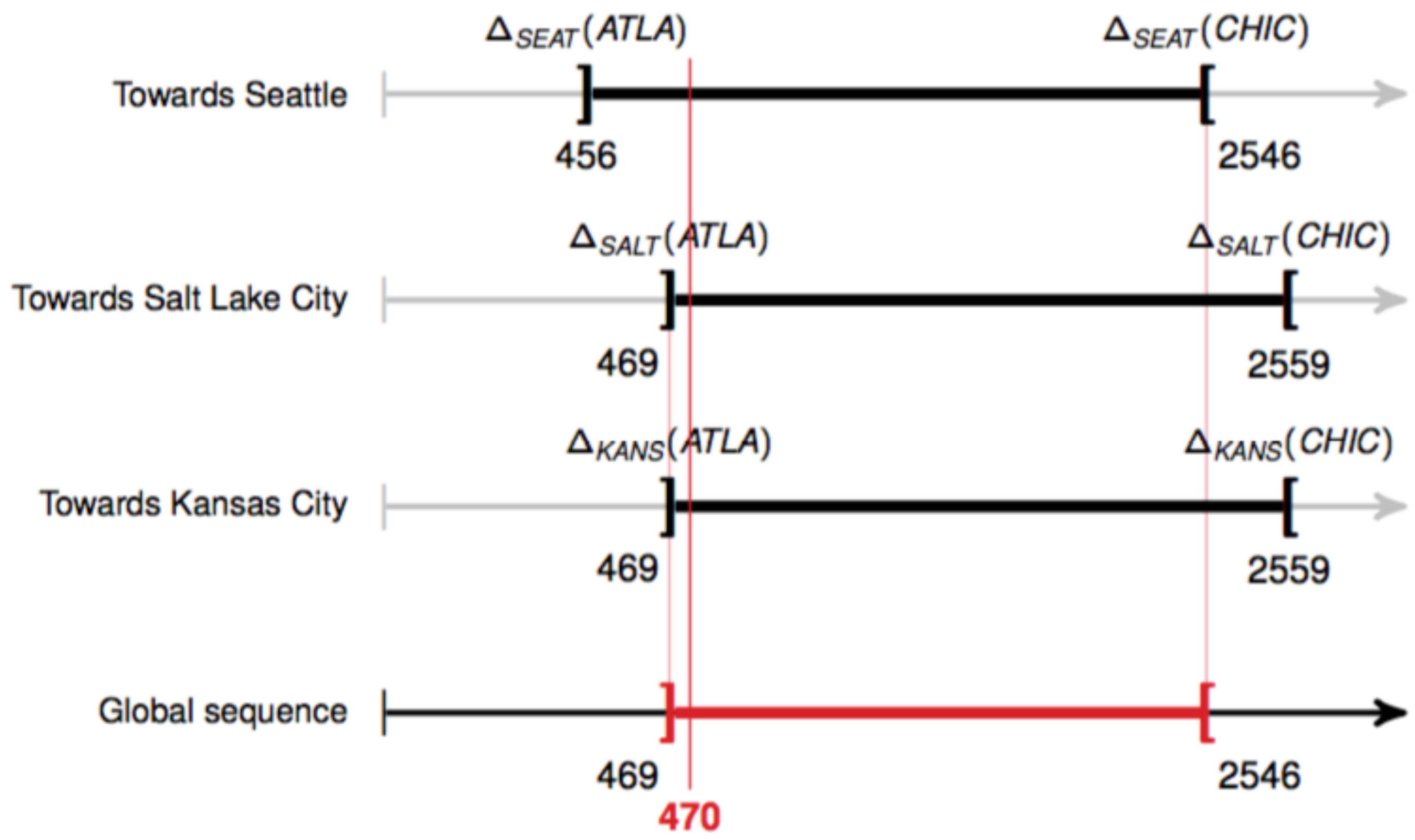
$e < 456$ Nothing happens...

$456 < e < 2546$ Atlanta reroutes.

$e > 2546$ Chicago reroutes.

Several destinations, several loops: how to treat them?

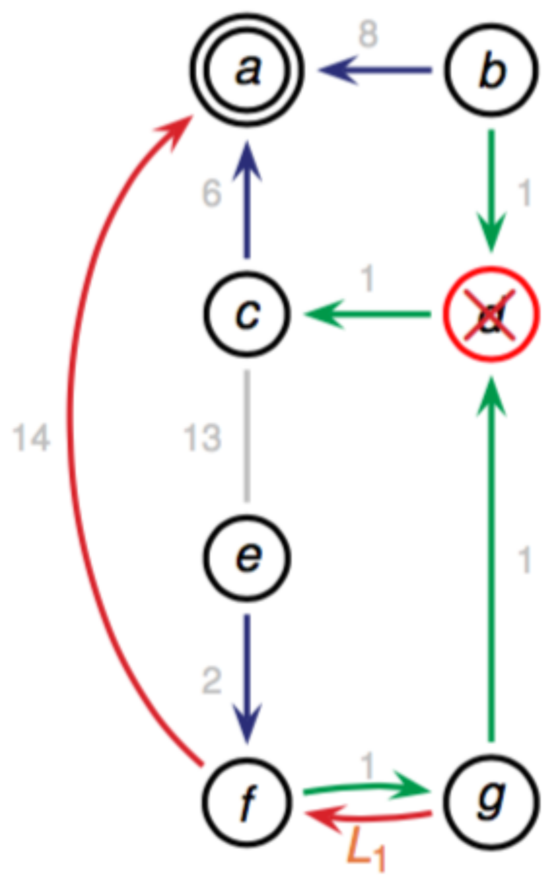
- **Intervals intersections for all destinations and loops**



Minimum loop-free sequence for the link $(CHIC, KANS)$: $S = \{470\}$

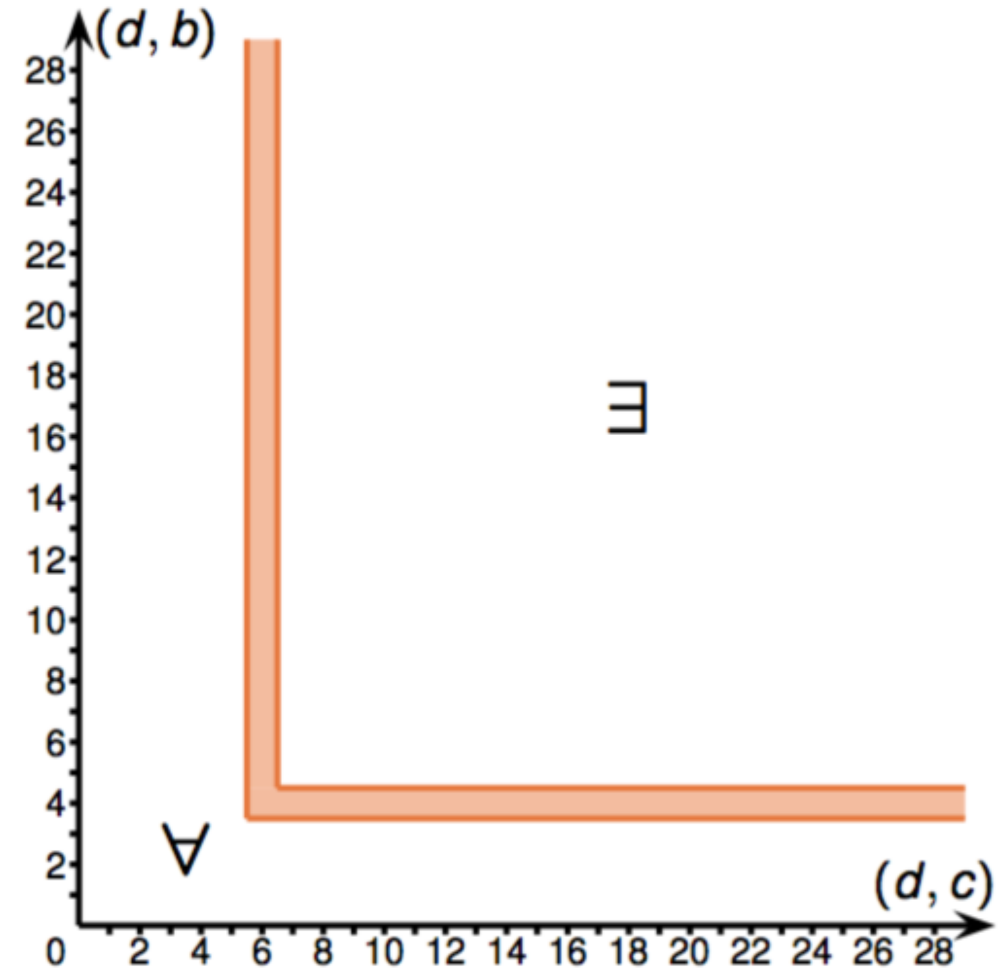
And with more dimensions?

- **What about a router-wide operation? Here with simply 2 outgoing links.**



$$\Delta_a^d(f) = \begin{pmatrix}5\\3\end{pmatrix}$$

$$\Delta_a^d(g) = \begin{pmatrix}7\\5\end{pmatrix}$$



Constraint c associated with a loop L

$$c := \left(\min_{\forall x \in L}(\Delta(x)), \max_{\forall x \in L}(\Delta(x))\right)$$

$$c_1 = \left(\begin{pmatrix}5\\3\end{pmatrix}, \begin{pmatrix}7\\5\end{pmatrix}\right)$$

A greedy algorithm that works in reverse fashion

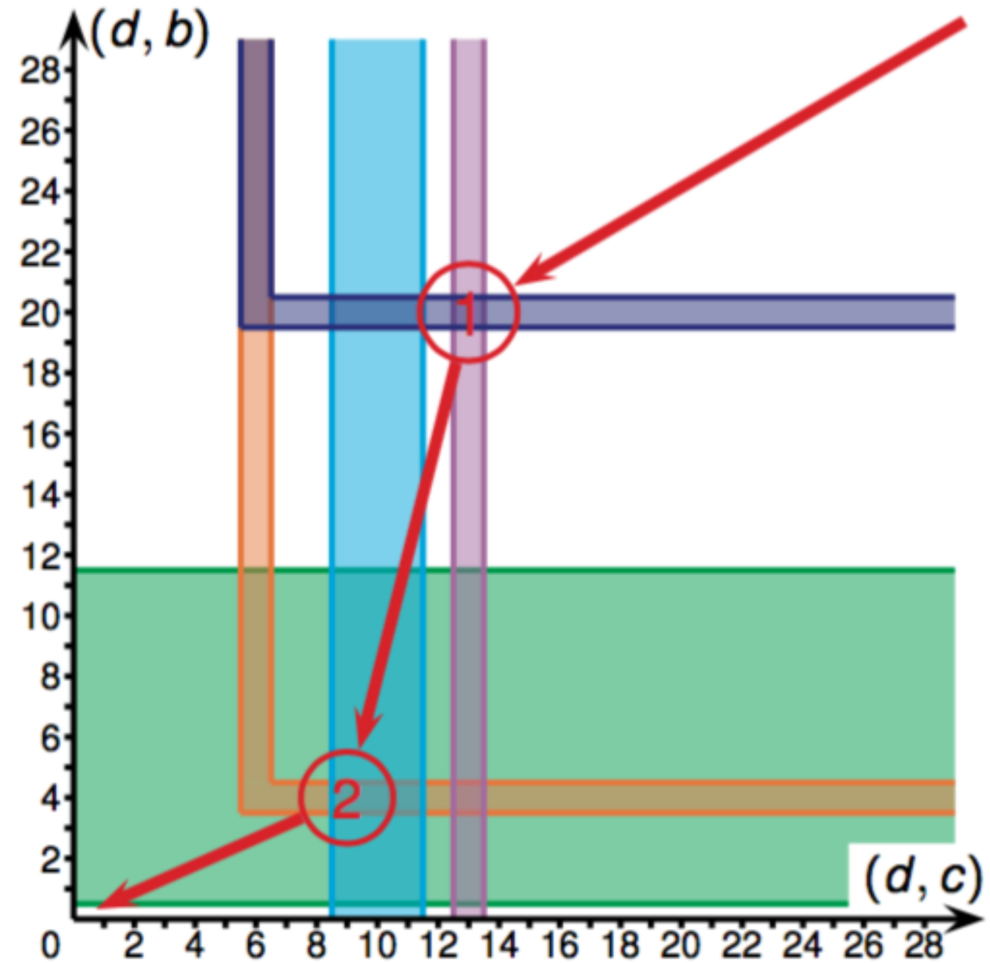
- **because it does not (always!) work in a forward fashion…**

Greedy Backward Algorithm (GBA)

At each step, retrieve the maximum value on each index among the lower bounds of the remaining constraints.

$$S_{GBA} = \left\{ \begin{pmatrix} 9 \\ 4 \end{pmatrix}, \begin{pmatrix} 13 \\ 20 \end{pmatrix} \right\}$$

c_1	c_2
c_3	c_5
c_4	



Theorem

Given a set of loop-constraints, *GBA* computes a minimal sequence of weight updates preventing all associated convergence loops.

GBA/D-CART summary

✓ Transient loops impact evaluation

▷ Loops do occur and impact the traffic in ISP networks

✓ Improvement of the existing approach

▷ Sequence minimality with polynomial time algorithms

▷ Efficient implementation

✓ Generalization to node-wide operations

▷ Practical solutions to deal with routing instabilities

- **=> To be tested in RENATER soon!**

http://icube-reseaux.unistra.fr/dcart

- **We get GEANT GN4 Open Call funds to make the story continue…**
  - …we need manpower for software development and platform management!
  - and more robust hardware (SD cards of R-PI are not!)
    - currently D-CART is down except the listener (it is running on a real server)
- **Are you interested in collaborating with us?**
  - or just discuss…
- **Have you any suggestions?**
  - or simply questions?

merindol@unistra.fr