cc-pipe: Breaking Systemic Bottlenecks in RPKI Data Supply Chain with Concurrent and Conflict-Free Pipelines

Chenhui Yu, Yanbiao Li*, Hui Zou, Yuxuan Chen, Shiyi Liu, Gaogang Xie*

中国科学院
计算机网络信息中心
Computer Network Information Center,
Chinese Academy of Sciences

中国科学院大学
University of Chinese Academy of Sciences

BGP is important yet vulnerable

- Border Gateway Protocol (BGP) is one of the key building blocks of the global Internet.
- However, BGP lacks built-in security protection and thus is vulnerable to prefix hijacks.

Cloudflare DNS Resolver Hit by BGP Hijack

The free Cloudflare DNS resolver service 1.1.1.1 was hit by a pair of simultaneous BGP attacks, showing that BGP is vulnerable even to accidental attacks.

by Paul Shread — July 6, 2024 Reading Time: 5 mins read

KLAYswap

KLAYSWAP|KLAYSWAP-BGP-HIJACK

Catalin Cimpanu

February 14th, 2022

News Cybercrime Technology

KlaySwap crypto users lose funds after BGP hijack

Hackers have stolen roughly $1.9 million from South Korean cryptocurrency platform KLAYswap after they pulled off a rare and clever BGP hijack against the server infrastructure of one of the platform's providers.

The BGP hijack—which is the equivalent of hackers hijacking internet routes to bring users on malicious sites instead of legitimate ones—hit KakaoTalk, an instant messaging platform popular in South Korea.

Testing mistake triggered Telstra route 'hijacks'

By Juha Saarinen
Oct 5 2020
7:33AM

5 Comments

RELATED ARTICLES

Mandiant names APT43 group as North Korean operation

TIO seeks digital platform role as scams proliferate

Microsoft introduces AI-powered cyber security assistant

Gov kicks off mobile audit process

Routing mishaps difficult to prevent.

An erroneous bulk upload of static routes to a Telstra production network edge router was the cause of last Wednesday's internet-wide service disruption that saw data traffic take a long detour via Australia, causing performance degradation for other providers in the process.

Telstra senior network engineer Mark Duffell apologised for the error, which meant that 500 internet protocol version 4 (IPv4) prefixes, or subnetworks, were advertised as belonging to Telstra.

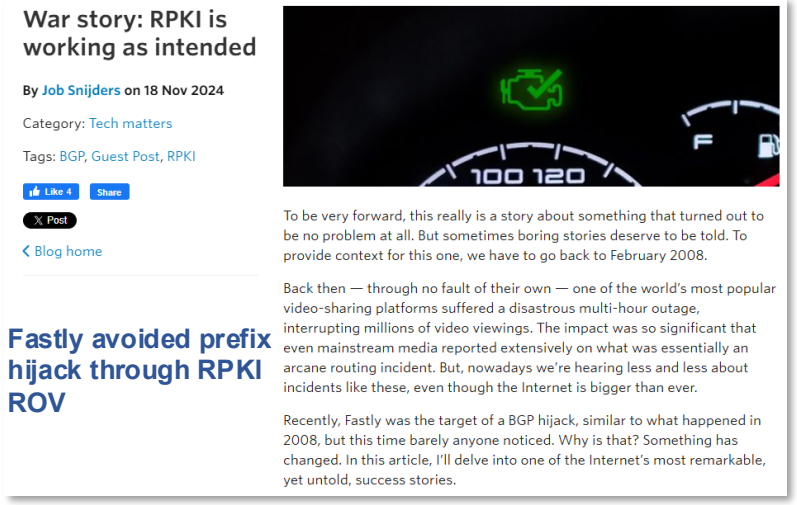
The technical error occurred as part of post-verification testing to address a software bug in the Telstra Internet Direct provisioning tools.

After the incorrect configuration was deployed to a single edge router the hundreds of IPv4 prefixes were announced to the global internet through the border gateway protocol (BGP) that supplies route information for network providers.

RPKI: Trusted Database for BGP

- **Resource Certificate (RC)**
  - Cryptographically delegates IP resources.
- **Route Origin Authorization (ROA)**
  - Cryptographically binds an AS with the prefix(es) it is authorized to advertise in BGP.
- **Route Origin Validation (ROV)**
  - Validates BGP routes using Validated ROA Payloads (VRPs).

[1]: https://blog.apnic.net/2024/11/18/war-story-rpki-is-working-as-intended/


War story: RPKI is working as intended

By Job Snijders on 18 Nov 2024

Category: Tech matters

Tags: BGP, Guest Post, RPKI

Like 4 Share

Post

Blog home

Fastly avoided prefix hijack through RPKI ROV

To be very forward, this really is a story about something that turned out to be no problem at all. But sometimes boring stories deserve to be told. To provide context for this one, we have to go back to February 2008.

Back then — through no fault of their own — one of the world's most popular video-sharing platforms suffered a disastrous multi-hour outage, interrupting millions of video viewings. The impact was so significant that even mainstream media reported extensively on what was essentially an arcane routing incident. But, nowadays we're hearing less and less about incidents like these, even though the Internet is bigger than ever.

Recently, Fastly was the target of a BGP hijack, similar to what happened in 2008, but this time barely anyone noticed. Why is that? Something has changed. In this article, I'll delve into one of the Internet's most remarkable, yet untold, success stories.

[2]: https://blog.cloudflare.com/cloudflare-1111-incident-on-june-27-2024/


Cloudflare 1.1.1.1 incident on June 27, 2024

2024-07-04

12 min read

This post is also available in 简体中文, Français, Deutsch, 日本語, 한국어, Español and 繁體中文.

[3]: https://internet2.edu/what-the-research-education-community-learned-from-three-impactful-routing-security-incidents-in-2024/


ADVANCED NETWORKING Security

11 September 2024

What the Research & Education Community Learned From Three Impactful Routing Security Incidents in 2024

Subscribe for more like this →

SHARE

By Steven Wallace - Director, Internet2 Routing Integrity

Estimated reading time: 4 minutes

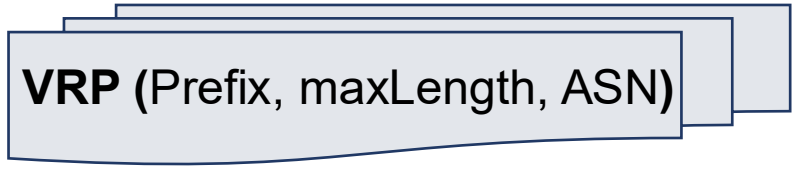
Countries with BGP autonomous systems represented in the GREN

RPKI has proved its effectiveness in preventing major real-world attacks

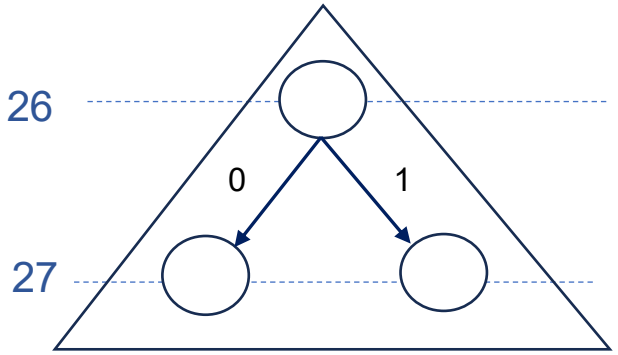
RPKI ROA

ROA (Route Origin Authorization) ➔ [Cryptographic Validation] ➔ **VRP** (Validated ROA Payload)

- ROA/VRP binds an AS with the prefix(es) it is authorized to advertise in BGP.



e.g. <192.0.2.64/26, 27,AS V>



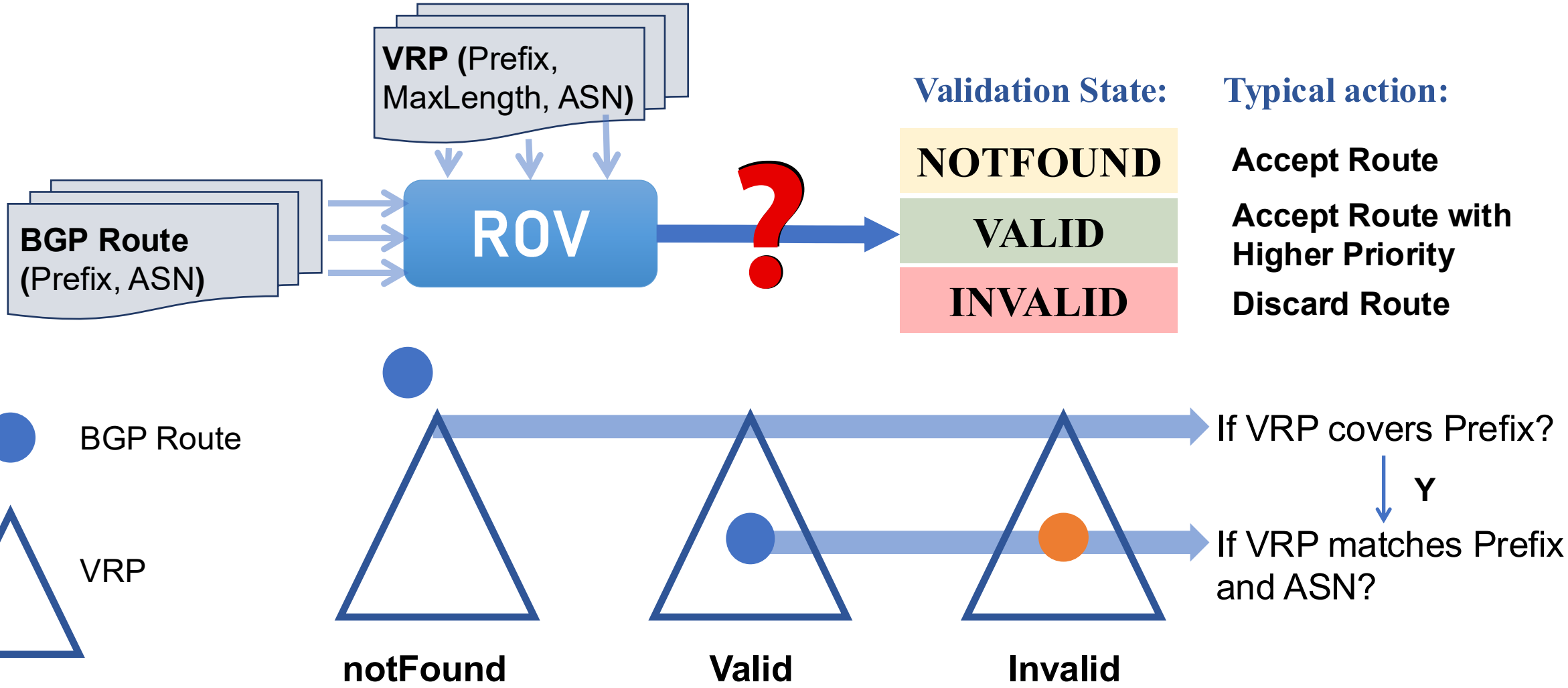
L26 192.0.2.64/26

L27 192.0.2.64/27
192.0.2.96/27

ASV can announce these Prefixes legitimately

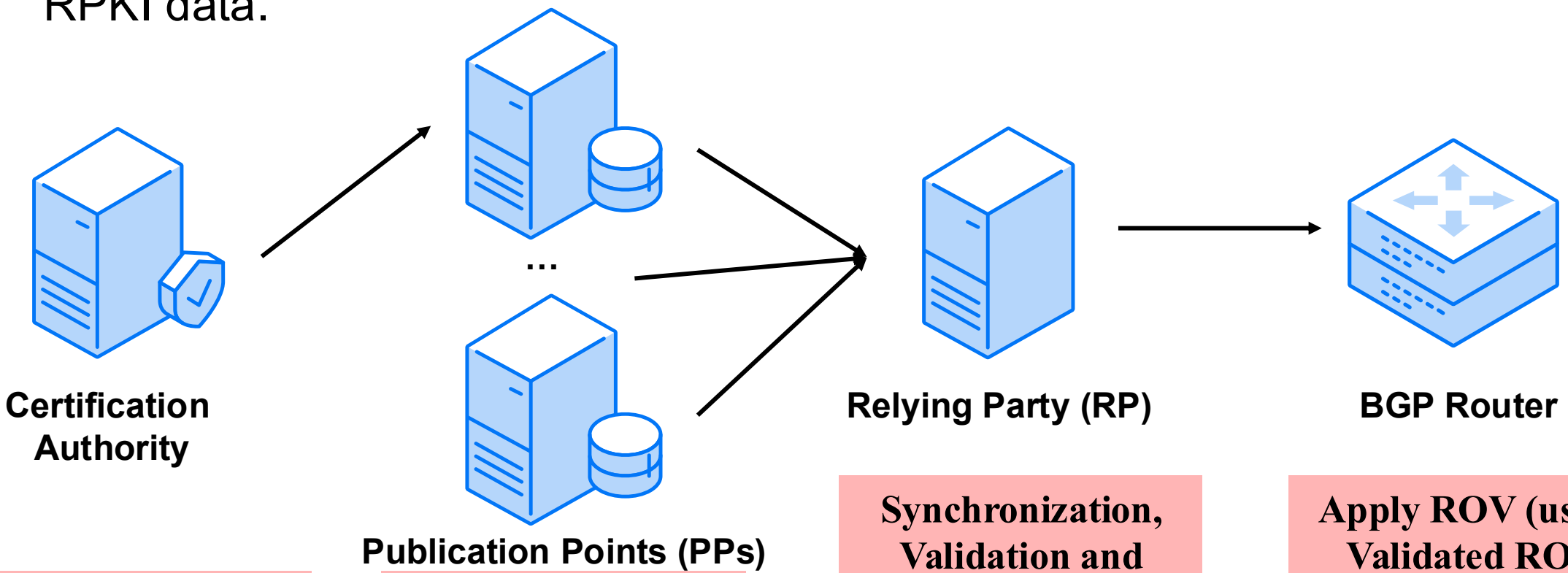
RPKI ROV

- Validates BGP routes by verifying the origin of route prefixes using VRPs.



RPKI Data Supply Chain

- ROAs are delivered to routers via RPKI data supply chain
- The efficiency of RPKI data supply chain affects the freshness of ROV side RPKI data.



Concerns about RPKI Data Supply Chain Efficiency

Problem: high latency of RPKI data supply chain affects the data freshness, thus undermining RPKI's security guarantees and impeding network operations.

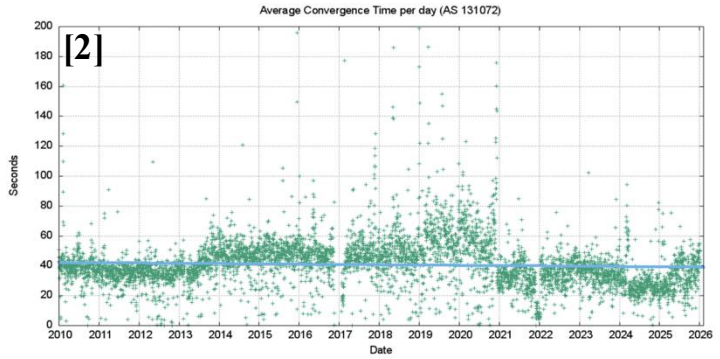
RPKI data supply chain latency: over 15 minutes in median

[1]	Sign*	NotBefore*	Publication†	Relying Party†	BGP‡
AFRINIC	0 (0)	0 (0)	3 (2)	14 (13)	15 (16)
APNIC	10 (13)	10 (13)	14 (16)	34 (38)	26 (28)
ARIN	- (-)	- (-)	69 (97)	81 (109)	95 (143)
LACNIC	0 (0)	- (-)	54 (32)	66 (42)	51 (34)
RIPE	0 (0)	0 (0)	4 (4)	14 (13)	18 (18)
After fix:					
ARIN	- (-)	- (-)	8 (9)	21 (22)	28 (23)

Relying Party(RP) is the bottleneck

vs

BGP convergence latency: less than 1 minute



Over 15x operational latency

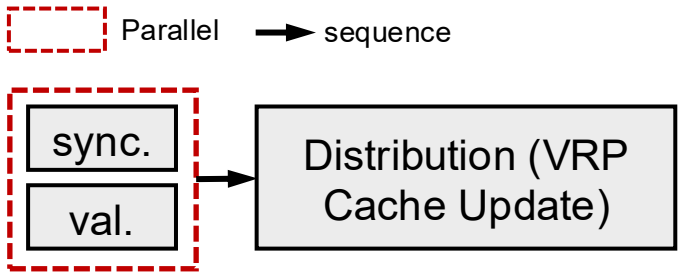
As RPKI adoption grows, the data volume and the number of publication points will increase, making **this issue more severe.**

[1]: Fontugne, Romain, et al. "Rpki time-of-flight: Tracking delays in the management, control, and data planes." International Conference on Passive and Active Network Measurement. Cham: Springer Nature Switzerland, 2023.

[2]: https://blog.apnic.net/2026/01/09/bgp-updates-in-2025/

Relying Party Working Paradigm

RP periodically performs **validation run** to synchronize data from all PPs and validate all data, then update its cache and distribute data to BGP router



RP's *monolithic paradigm: "wait-for-all" sequential order* (mandated by IETF RFC)

- **Pros: good consistency**
  - prevents intermediate changes to a route's validation state that lead to unsafe routing or traffic loss.
- **Cons: high latency**
  - straggler effect causes high blocking latency (the duration that ready-to-use data is unnecessarily delayed before cache update)

Blocking latency dominates the validation run latency (averagely 50%-84% for different RPs)

Prior works: Reduce RP Workload

Reduce Synchronization Workload

- Data compression [INFOCOM22]
- New Publication Point Architecture [NDSS24]
- Lighter encoding scheme [NDSS26]

Reduce Validation Workload

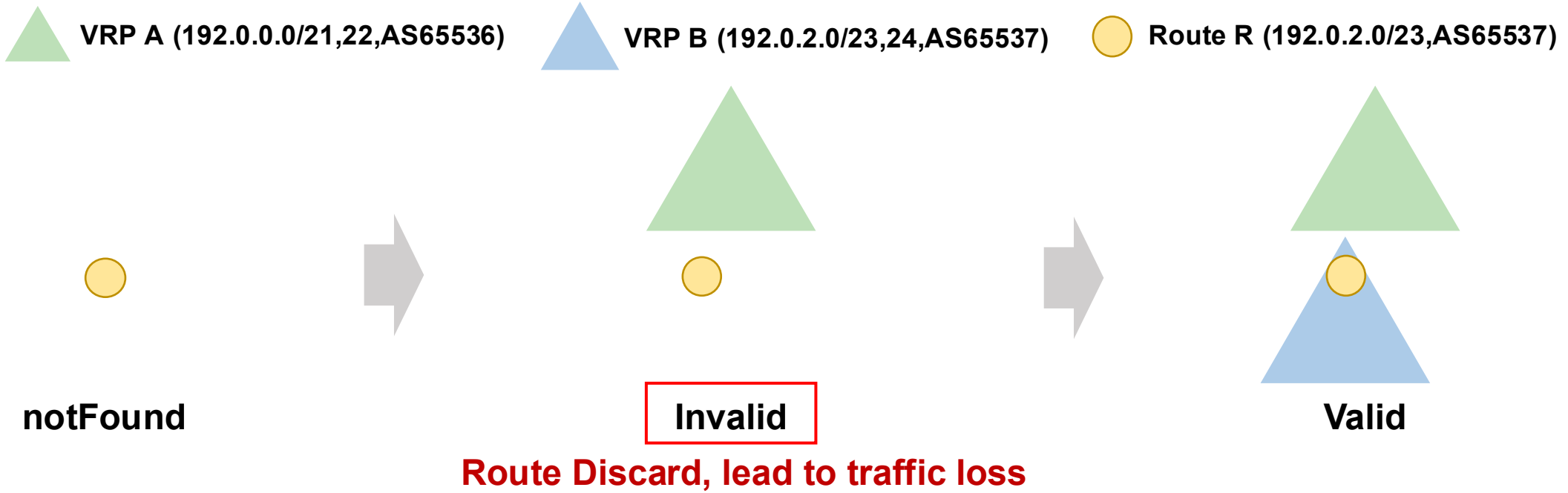
- On demand Validation Algorithm[IETF draft]
- Pruning RPKI data [NDSS26]

Limitation: Accept the validation run's monolithic paradigm; their performance remains constrained by the inherent "wait-for-all" constraint

[INFOCOM22] Li Y, Zou H, Chen Y, et al. The hanging ROA: A secure and scalable encoding scheme for route origin authorization
[NDSS24] Su Y, Li D, Chen L, et al. dRR: A Decentralized, Scalable, and Auditable Architecture for RPKI Repository
[NDSS26] Schulmann, Haya, and Niklas Vogel. Pruning the Tree: Rethinking RPKI Architecture from the Ground up
[IETF draft] https://datatracker.ietf.org/doc/draft-madi-sidrops-partial-validation/

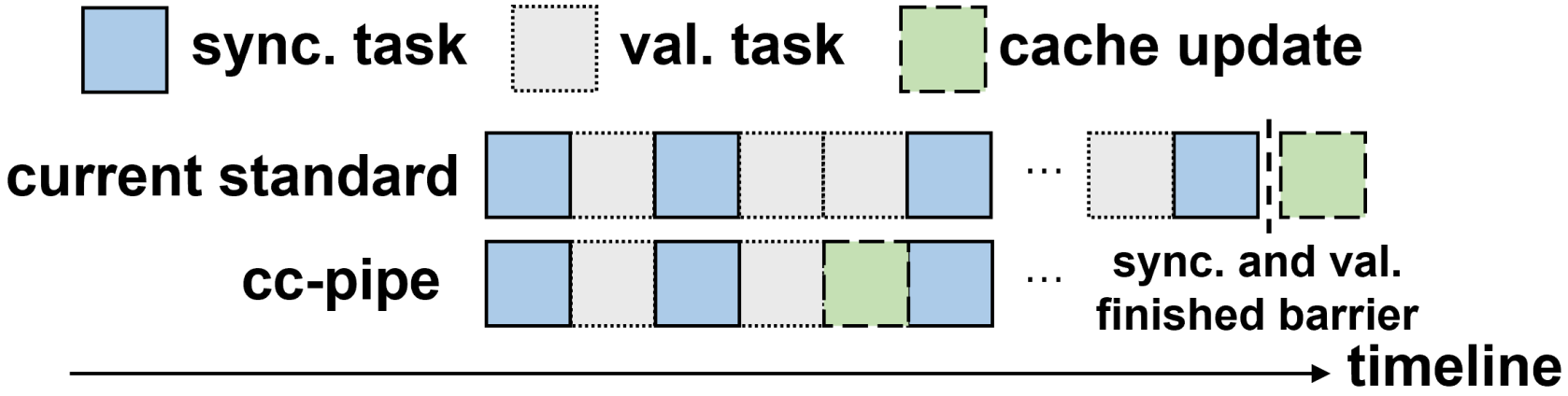
Strawman Solution: Naïve Pipeline

- **A Natural Idea:** Distribute VRP updates to routers immediately as soon as individual RPKI objects are validated, bypassing the global "wait-for-all" barrier.
- **The Fatal Flaw:** Ignoring dependencies between updates leads to **Inconsistent ROV state**, leading to **unsafe routing or traffic loss**.



Our Approach

- **The Bottleneck:** The "wait-for-all" barrier is the root cause of severe RP processing latency.
- **The Dilemma:** A naïve pipeline accelerates processing but trade consistency, lead to unsafe routing or traffic loss.
- **Our Approach:** Break the barrier with a **Concurrent and Conflict-Free Pipeline (cc-pipe)** that guarantees **routing consistency**.



Roadmap: Achieving Concurrent & Conflict-Free

1. how to incrementally calculate VRP updates?

By calculating “local view” VRP updates per Resource Certificate and maintain the “global view” .

Incremental VRP Maintenance

Local View Update Generation (per RC)

- Calculates the VRP update between current and previous VRP sets at the **Resource Certificate level**.

$$\mathrm{VRP}_{\mathrm{prev}} \triangle \mathrm{VRP}_{\mathrm{curr}} \rightarrow \text{Local View Update}$$

Global State Management

- Maintains **global correctness** when multiple RCs have same VRP.
  - Current VRP Snapshot is all VRP that count larger than 0
  - Compare different version's VRP Snapshot to get VRP delta

VRP	Global Count
192.0.0.0/21,22, AS65536	0
192.0.0.0/21,22, AS65536	1
…	…

Roadmap: Achieving Concurrent & Conflict-Free

1. how to incrementally calculate VRP updates?

By calculating “local view” VRP updates per Resource Certificate and maintain the “global view” .

2. how to distribute VRP updates in a conflict-free manner?

By utilizing the Resource Containment Invariant: the resources allocated to a child RC and the IP prefixes issued in a ROA are subsets of the resources held by the parent RC.

Principle: Atomic Commitment

- **Definition of Conflict:** Two VRP updates v_i, v_j conflict if their **prefixes overlap**, which means they can affect the same route.

- **The Theoretical Constraint:** To maintain consistency, conflicting updates must follow strict ordering rules (detailed in paper). And distributing conflicting updates simultaneously always meets ordering rules.

- **Our Design Choice:** Instead of implementing a complex, fine-grained scheduler, cc-pipe opts for **Atomic Commitment—grouping conflicting updates** to distribute them simultaneously.

Measurement Results: The **vast majority** of updates are **conflict-free** (68.6%), and most conflicts (98.4%) are localized within the same Resource Certificate (RC).

Key Idea: Predictive Conflict Resolution

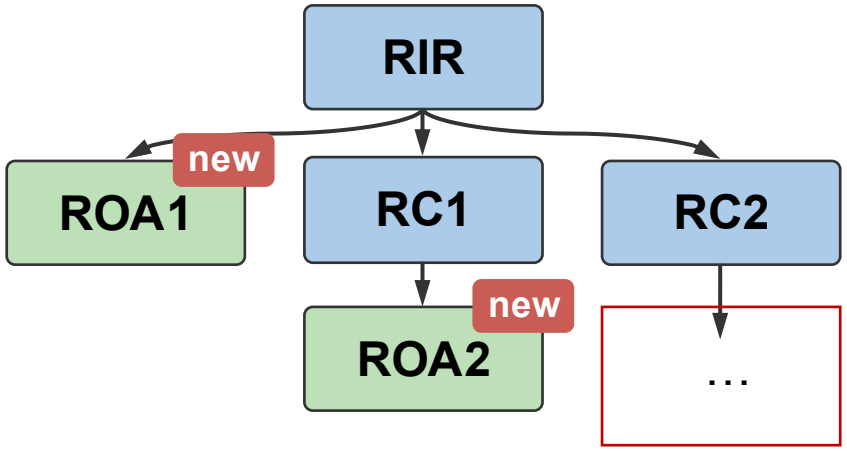
Identify Acquired Conflicts (VRP-VRP)

- **Prefix Overlap Detection:** Identify conflicts between already obtained VRP updates if their IP prefixes overlap.

Key Idea: Predictive Conflict Resolution

Detect Future Conflicts (RC-VRP)

- **The Challenge:** How to detect conflicts with VRP update that hasn't been obtained yet?
- **The Solution:** Leverage the **Resource Containment Invariant**.
- **Predictive Locking:** Use the root resource certificate of a **unprocessed certificate subtree** as a maximum scope to lock potential future VRP updates.

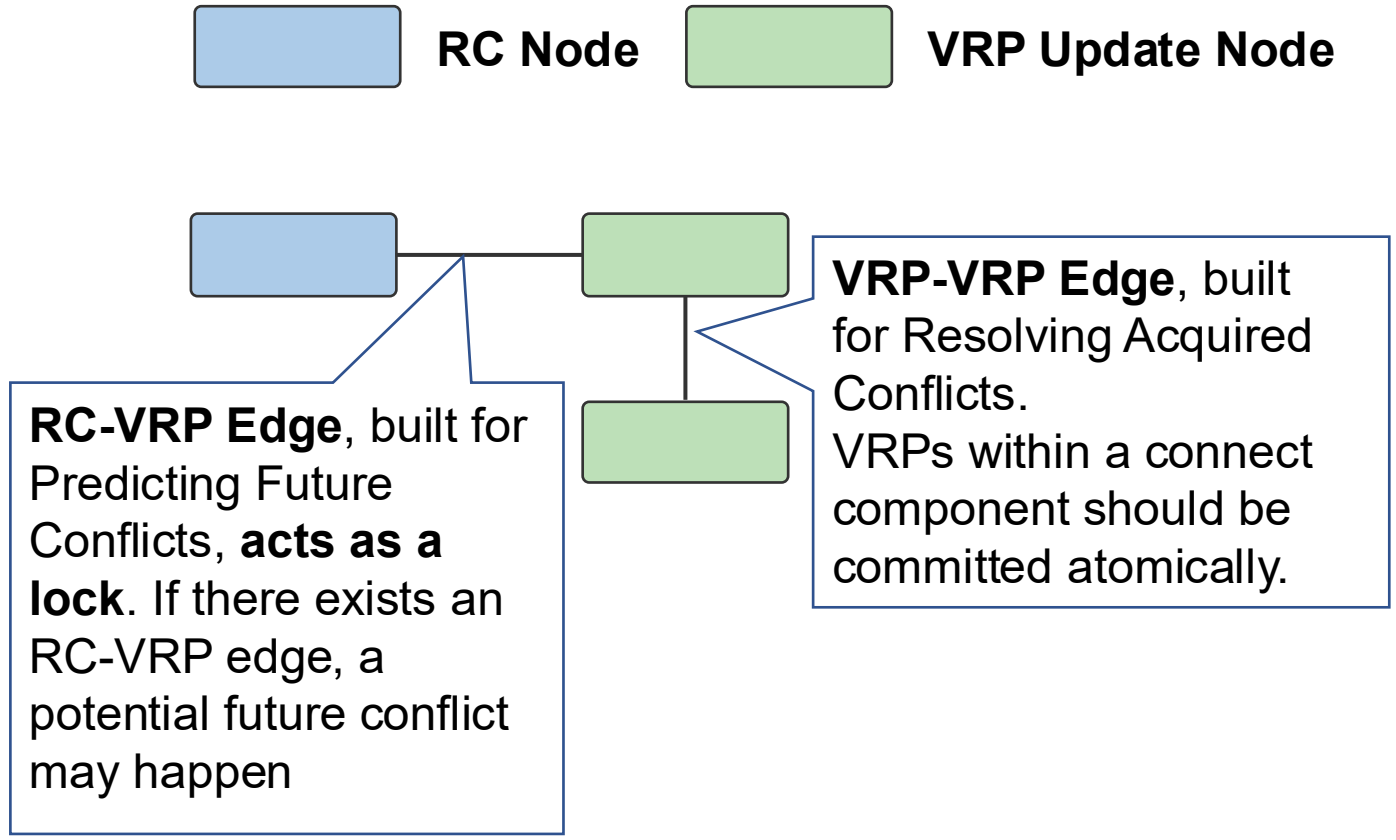


Resource Containment Invariant

No matter how ROAs change in the unprocessed certificate subtree, their IP prefixes are strictly **bounded by the subtree's root RC's resources.**

Conflict Graph: Basic Component

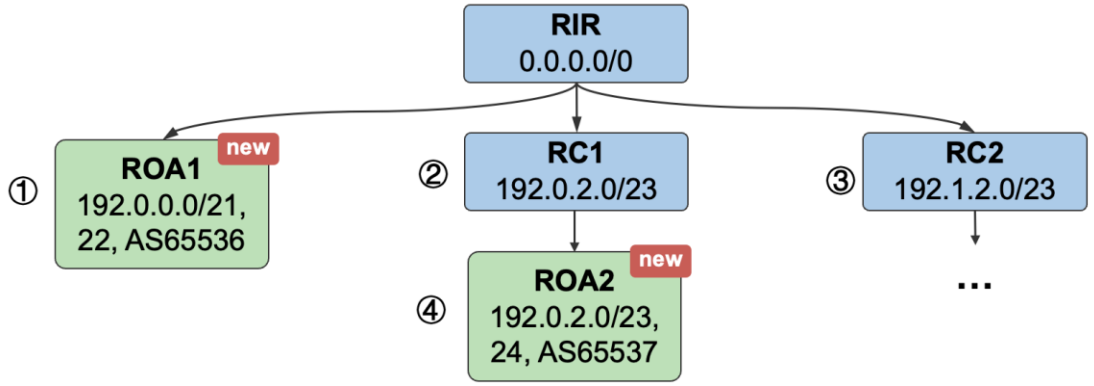
- **Nodes:**
  - **VRP Update Node**
    - A local view VRP update
  - **RC Node:**
    - Predictive lock for a unprocessed subtree.
- **Edges:**
  - **RC-VRP Edge:**
    - Connects a predictive lock to a VRP update.
  - **VRP-VRP Edge**
    - Connects two VRP update nodes.



Conflict Graph: Building and Usage

1. Initialization

- Initialize with root RC nodes (Trust Anchors)



An RPKI Certificate Tree Example

Initial

Conflict Graph

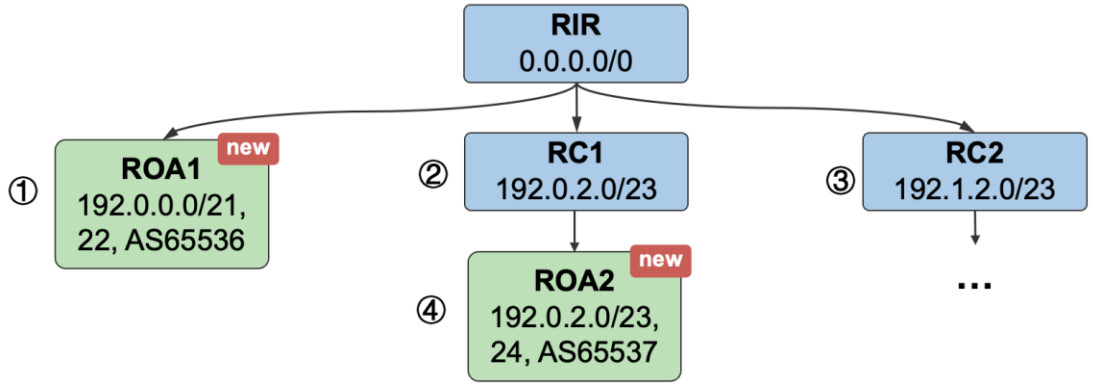
Conflict Graph: Building and Usage

1. Initialization

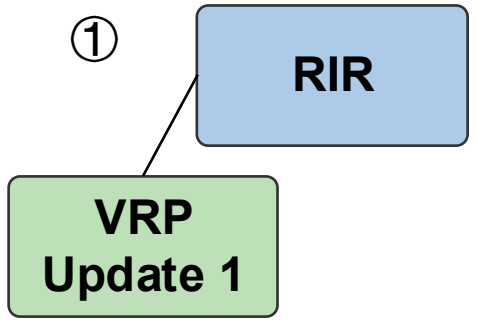
- Initialize with root RC nodes (Trust Anchors)

2. Graph Expansion

- Insert new VRP/RC nodes and build conflict edges as data arrives.



An RPKI Certificate Tree Example



Conflict Graph

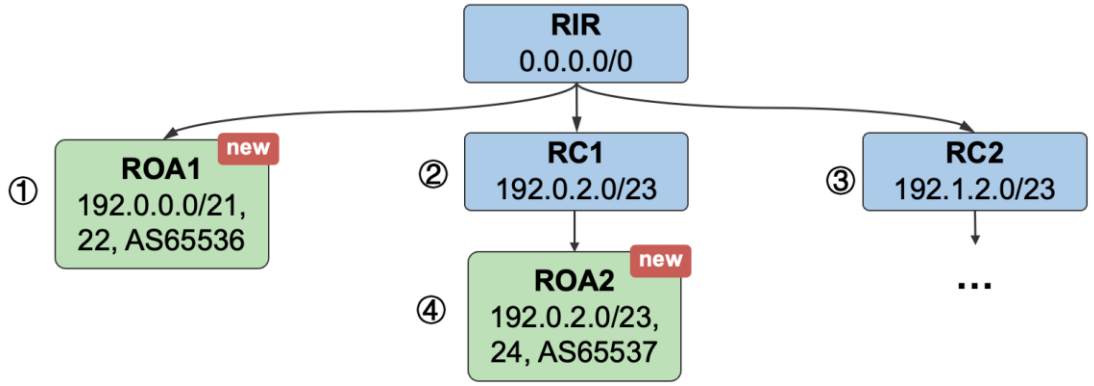
Conflict Graph: Building and Usage

1. Initialization

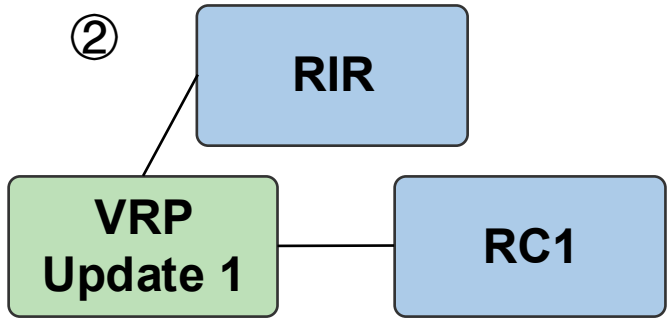
- Initialize with root RC nodes (Trust Anchors)

2. Graph Expansion

- Insert new VRP/RC nodes and build conflict edges as data arrives.



An RPKI Certificate Tree Example



Conflict Graph

Conflict Graph: Building and Usage

1. Initialization

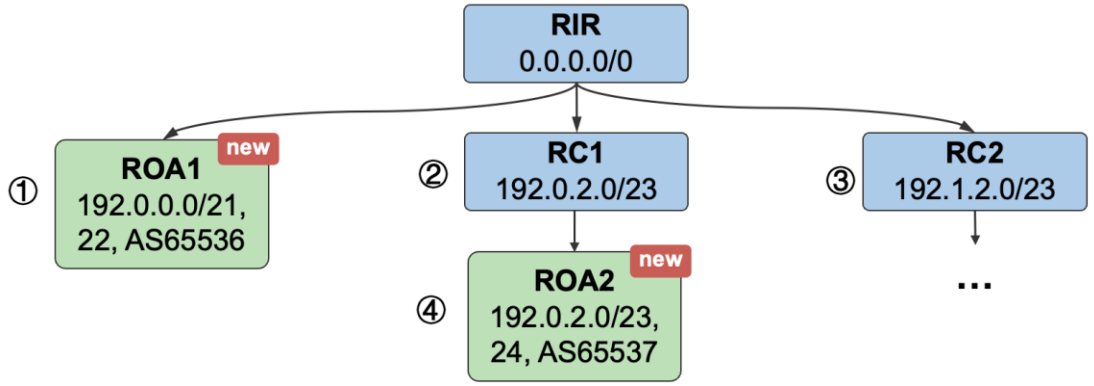
- Initialize with root RC nodes (Trust Anchors)

2. Graph Expansion

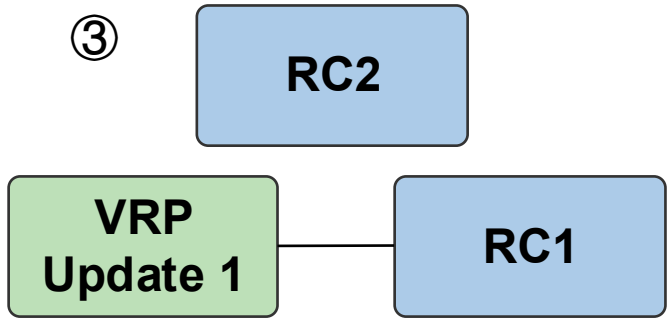
- Insert new VRP/RC nodes and build conflict edges as data arrives.

3. Graph Contraction

- Remove an RC node only after it is fully explored (all its direct descendant objects have been validated and the generated nodes are inserted).



An RPKI Certificate Tree Example



Conflict Graph

Conflict Graph: Building and Usage

1. Initialization

- Initialize with root RC nodes (Trust Anchors)

2. Graph Expansion

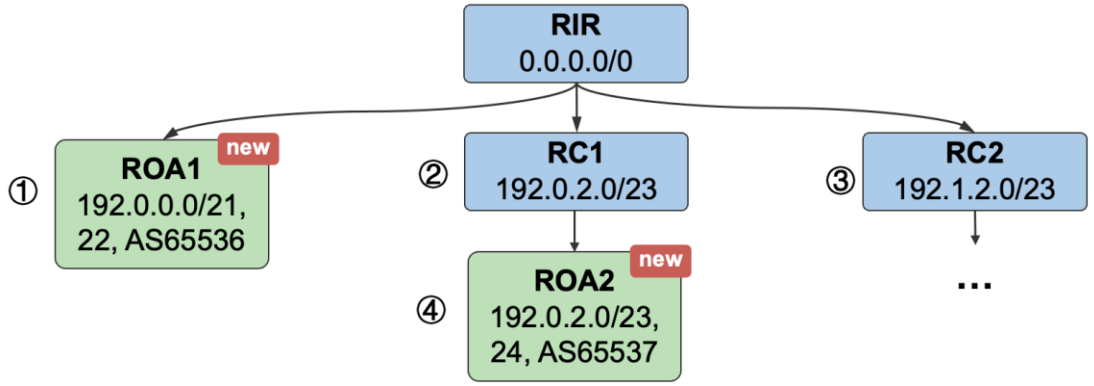
- Insert new VRP/RC nodes and build conflict edges as data arrives.

3. Graph Contraction

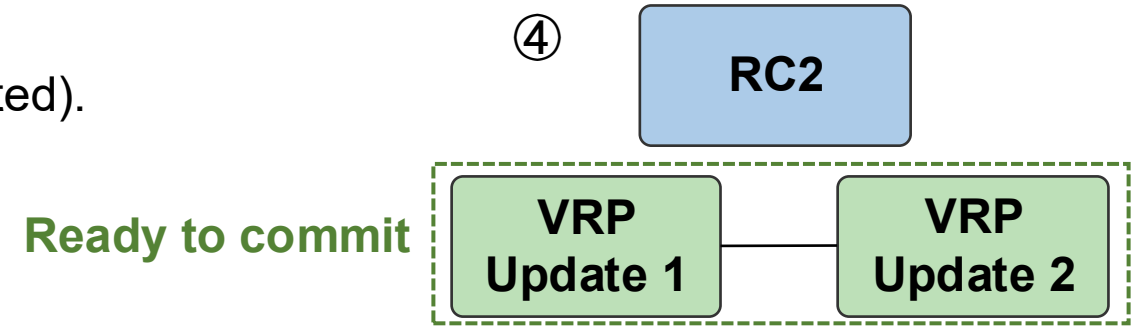
- Remove an RC node only after it is fully explored (all its direct descendant objects have been validated and the generated nodes are inserted).

4. Conflict-free Clustering

- When a connected component of conflict graph only contain VRP update nodes, they can be distributed simultaneously.



An RPKI Certificate Tree Example



Conflict Graph

Roadmap: Achieving Concurrent & Conflict-Free

1. how to incrementally calculate VRP updates?

By calculating “local view” VRP updates per Resource Certificate and maintain the “global view” .

2. how to distribute VRP updates in a conflict-free manner?

By utilizing the Resource Containment Invariant: the resources allocated to a child RC and the IP prefixes issued in a ROA are subsets of the resources held by the parent RC.

3. how to reduce router overhead?

By introducing aggregation interval and Validation-Run Centric VRP Cache.

RP-Router Communication

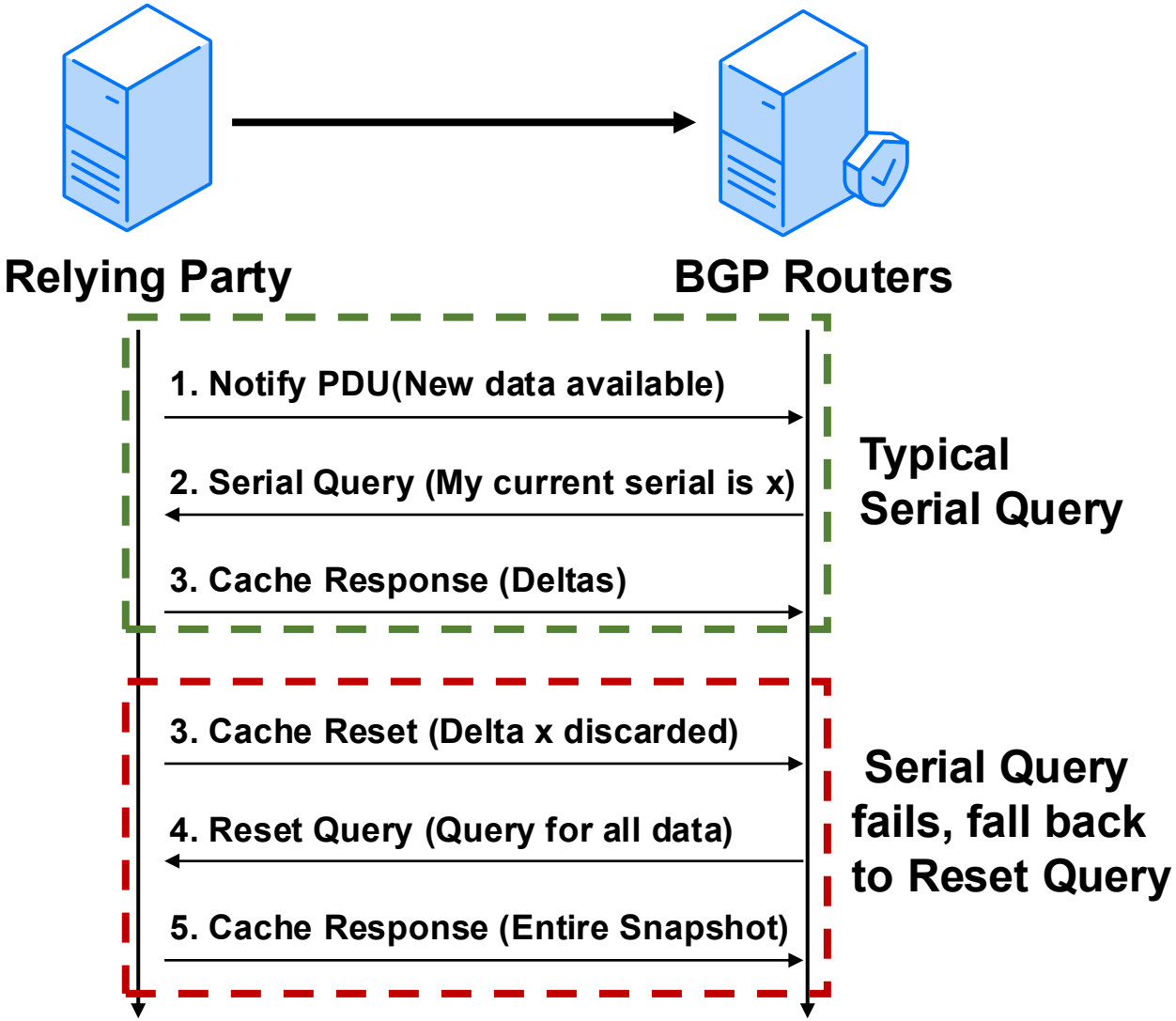
RP Side:

- **RP Notification:** RP notifies router when new data is available.

Router Side:

- **Serial Query:** only downloads what changed since its last update and validates affected routes. ***(Low Overhead)***

- **Reset Query:** download the entire routing snapshot and validate all route. ***(Massive CPU/Network Overhead)***



Control Router Overhead

Pipelining splits one delta per validation run into many fine-grained deltas.

Overhead from Frequent Notification

Problem: Pipelining creates **high-frequency** new data notifications.

Solution: Aggregation Interval

Updates *in* $[T, T + Interval] \rightarrow$ *Single Notify*

Balances VRP freshness with routing plane CPU load via tunable rate-limiting.

Overhead from from Serial Query Failures

Problem: VRP cache maintains the latest n deltas; rapidly generating deltas leads to a fallback to **Heavy Reset Queries**.

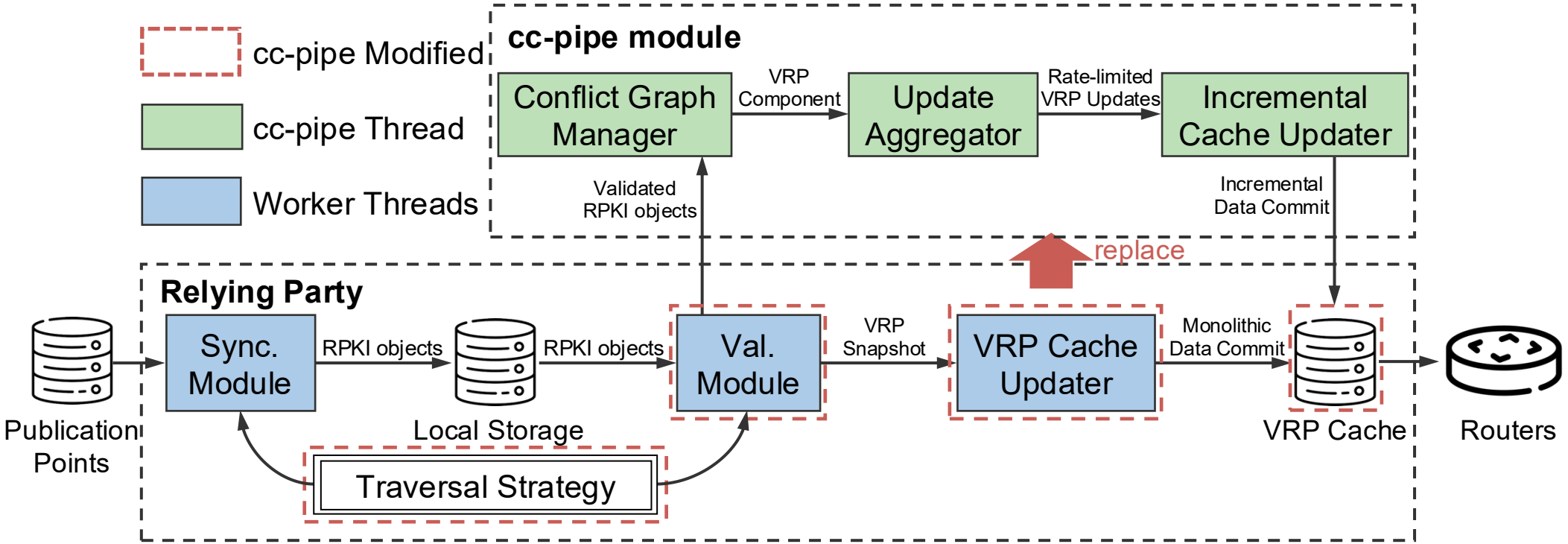
Solution: Validation-Run Centric Cache

- Shift from "Delta-counting" to "Run-tracking".
- Maintain all deltas from the latest n Validation Runs

Consistent History Depth

System Implementation

Reusable Library	Multi-RP Integration	Other Optimization
cc-pipe's core logic is implemented as a reuseable library.	integrated into Routinator(Rust) and FORT(C via FFI).	Modified FORT's certificate tree traversal strategy to a more pipeline friendly one



Evaluation Methodology

Evaluation Scenarios

- **Real-World Deployment:**
  - Latency/Overhead during live trial.
- **Future Deployment:**
  - Scalability test for future growth (100% deployment ratio and 10,000 Publication Points).
- **Slow PPs:**
  - Robustness against misbehaving (slow) Publication Points.

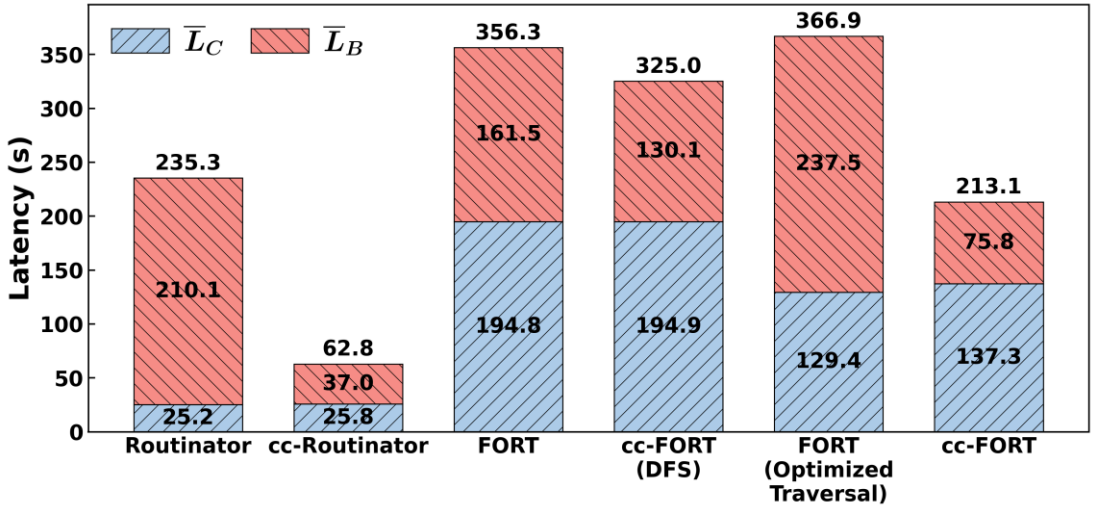
Data Sources

- **Real-World Deployment:**
  - One-month live trial (Aug 17 - Sep 17,2025) across global RPKI ecosystem.
- **Future Deployment and Slow PPs:**
  - Generated data

Compared Schemes

- **Stock Version RPs versus cc-pipe Integrated Version RPs**

Real-World deployment Scenario



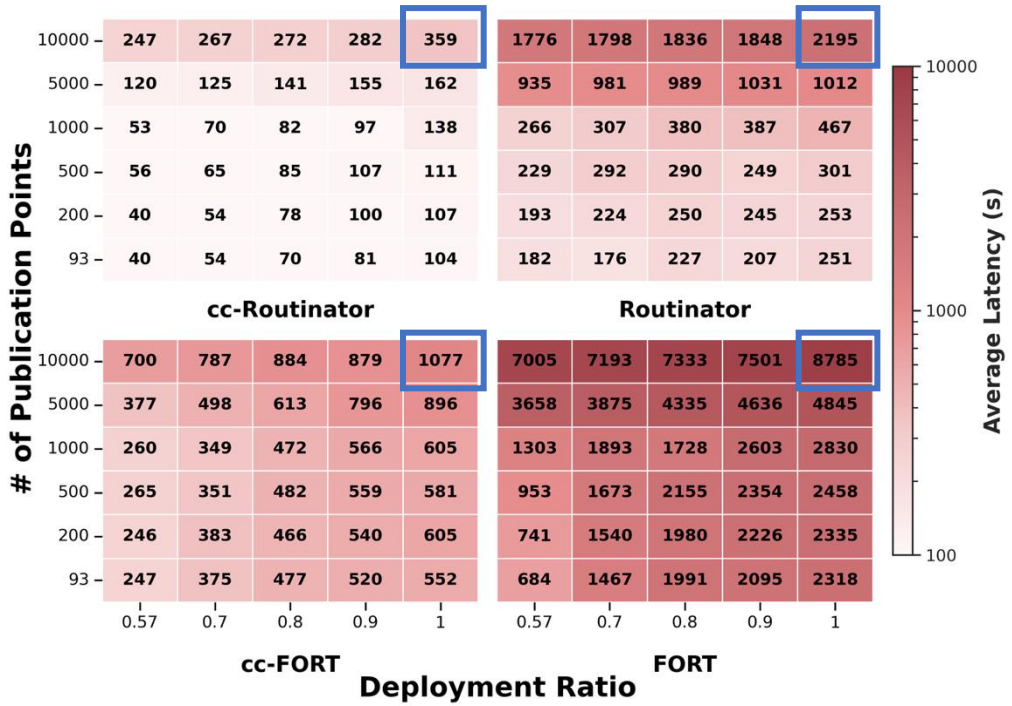
RP implementations	CPU (%)		Memory (MB)	
	avg.	peak	avg.	peak
Routinator	8.01	449.50	465	896
cc-Routinator	8.16	499.50	873.44	1,473.50
FORT	13.16	401.60	526.17	597.02
cc-FORT	14.24	458.50	775.20	1,075.73

On average 73.3% (41.9%) latency reduction for Routinator (FORT-Validator)

Modest RP resource usage increase, easily absorbed by modern server hardware.

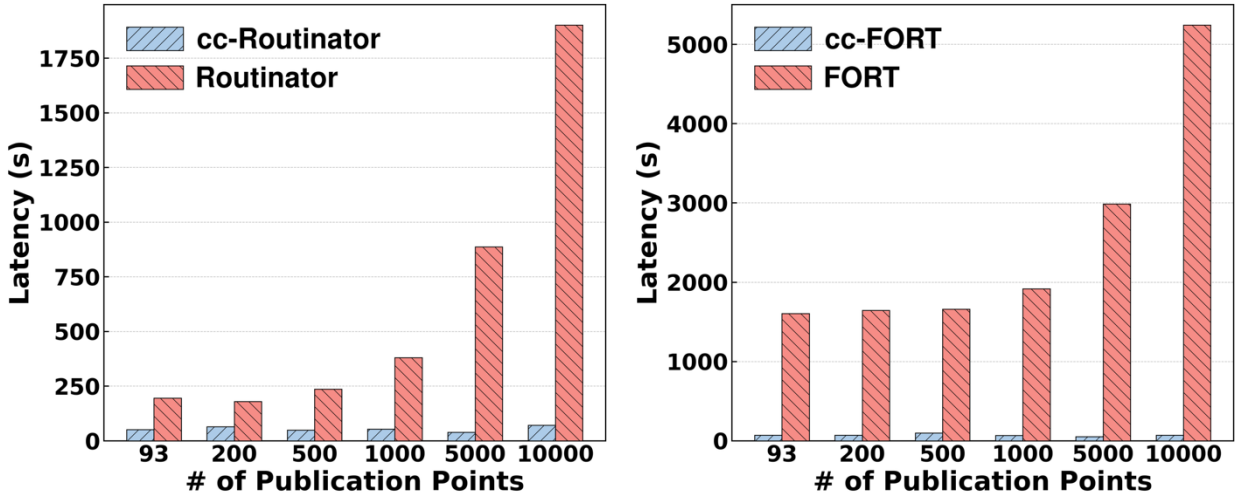
No statistical change in CPU load for BGP routers (BIRD/FRRouting)

Future deployment scenario



Total latency

At most over 2.1 hours latency reduction for the future workload



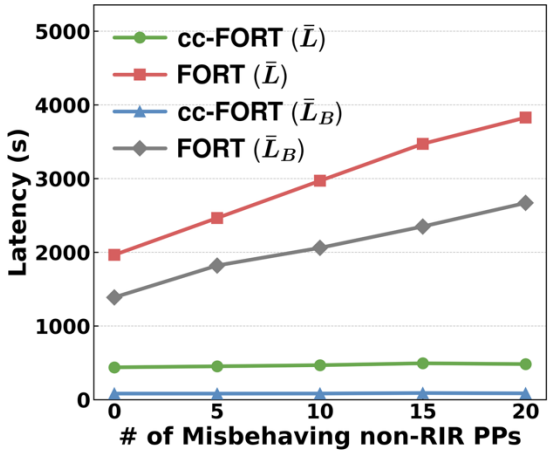
Blocking latency

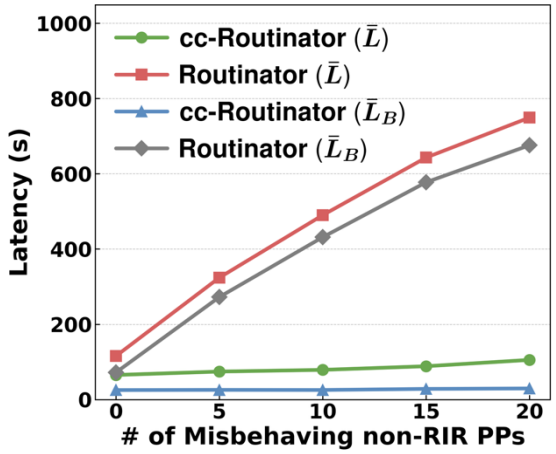
Massive performance gains are achieved by erasing structural blocking latency.

Slow Publication Points scenario

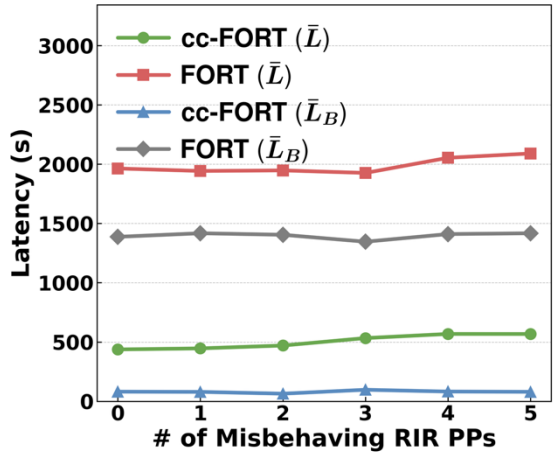
2 different sub scenarios: slow non-RIR PP and slow RIR PP (RIR PPs are maintained by RIRs and contains 'root' Resource Cert of RIRs)

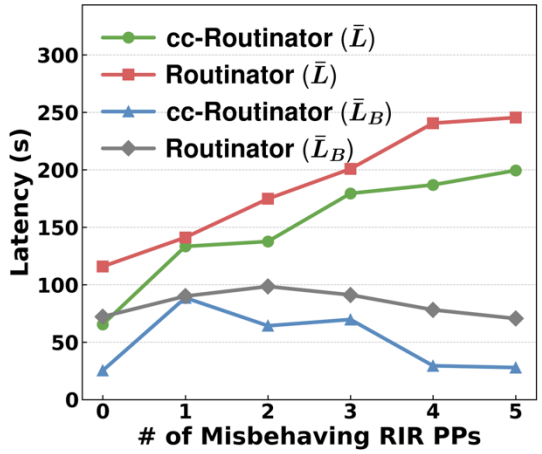
Slow Non-RIR PPs





Slow RIR PPs





Latency remains low and stable, while stock validators degrade by up to 6x.

Safely delays updates to enforce strict routing consistency, yet still outperforms stock baselines.

Conclusion: cc-pipe

- **Paradigm Shift**
  - **Breaks** the inefficient **"Wait-for-all" barrier** by introducing concurrent & conflict-free pipelines.
- **Higher Performance**
  - up to **73.3% average latency reduction**, superior scalability and robustness.
- **Immediate Deployment**
  - **fully compatible** with the current RPKI infrastructure, could be deployed by a software upgrade.

Thank you!

For more details, please read our paper:
cc-pipe: Breaking Systemic Bottlenecks in RPKI Data Supply Chain with Concurrent and Conflict-Free Pipelines

Contact: lybmath@cnic.cn

Code: https://github.com/FIRLab-CNIC/cc-pipe

中国科学院
计算机网络信息中心
Computer Network Information Center,
Chinese Academy of Sciences

中国科学院大学
University of Chinese Academy of Sciences