An Evaluation of IPFS As A distribution Mechanism for RPKI Repository

Dadepo Aderemi, Woudt van Steenbergen
Supervisor: Luuk Hendriks | NLnet Labs
July 2, 2020

IPFS Primer

What

- A **peer-to-peer distributed** file system that seeks to connect all computing devices with the same system of files. [7]

Why

- Distributed over centralised systems
- Efficient Data Transfer
- Resiliency
- Permanence

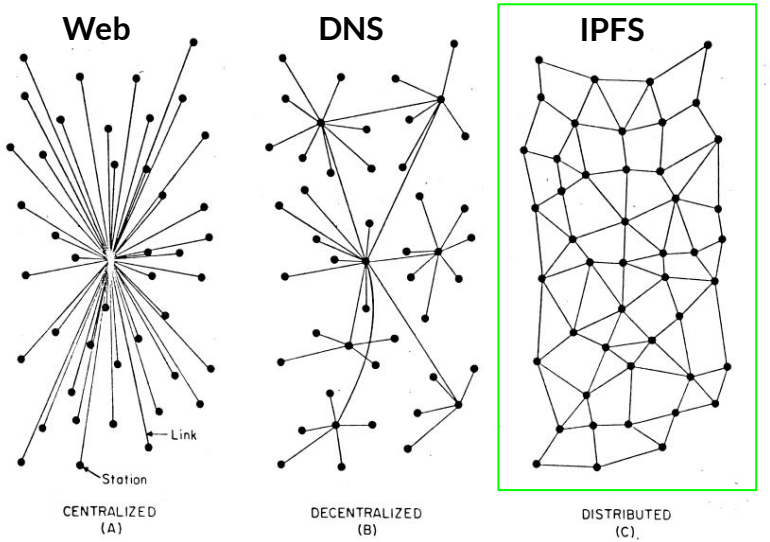


Fig. 1 - Centralized, Decentralized and Distributed Networks [1]

How

- Content Addressing
- InterPlanetary Linked Data (IPLD) formally Merkle DAG
- Distributed Hash Table (Kademlia)
- PKI based Identity

RPKI Primer

What

- Resource Public Key Infrastructure
- A PKI based approach to securing global internet routing
- Makes use of X509 certificates to prove ownership of Internet Number Resources (INR) - ASN, IPv4 and IPv6
- Owners of Internet Number Resources can make verifiable statement on how their resources can be used

Why

- Mechanism to make Internet routing more secure
- Border Gateway Protocol (BGP) has no inbuilt security
- Security is based on trust, which does not scale
- Leads to prevalent prefix hijacks and Misconfiguration mishaps

BGP without RPKI

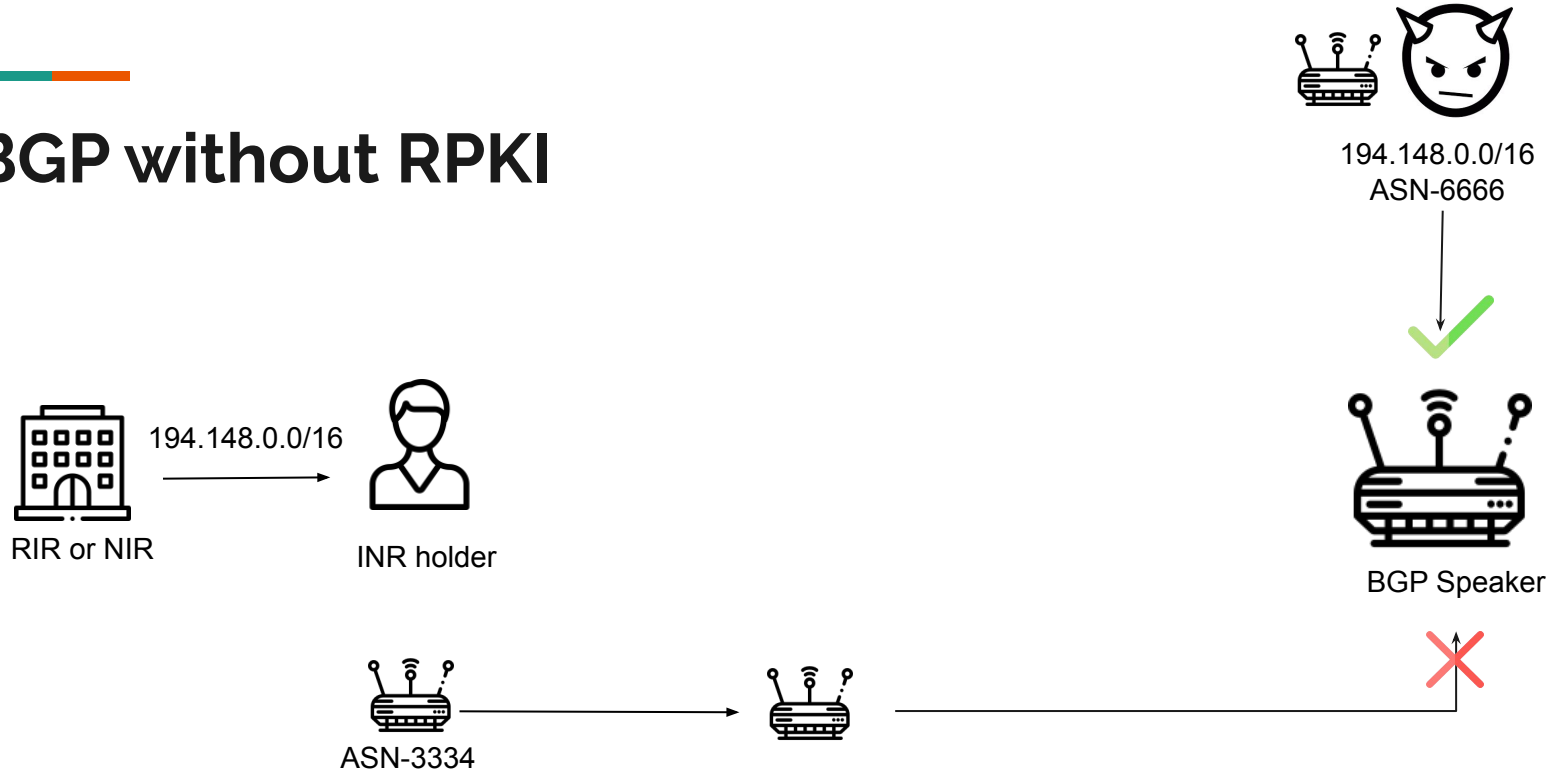


Fig. 2 - Prefix Hijacking in BGP

RPKI Primer

How

- Ties into the hierarchical resource allocation driven by Regional Internet Registry (RIR) and National Internet Registry (NIR)
- Resource is allocated to user, together with a resource certificate
- User creates Resource Origin Authorization (ROA)
- ROAs are published to publicly available repositories
- Relying Party (RP) downloads and creates Validated ROA Payload (VRP)
- BGP speakers uses VRP to make routing decision

BGP with RPKI

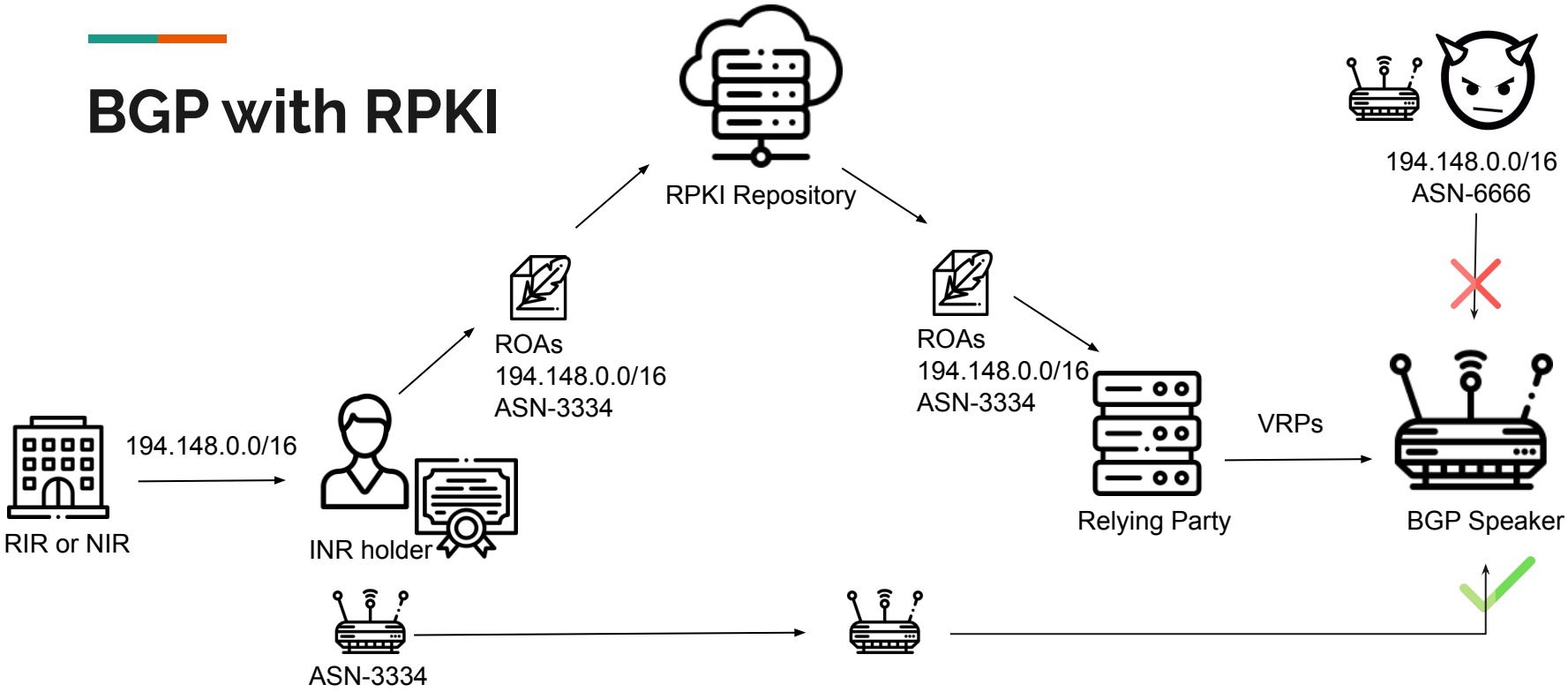


Fig. 3 - Preventing prefix hijacking with RPKI

RPKI Repository

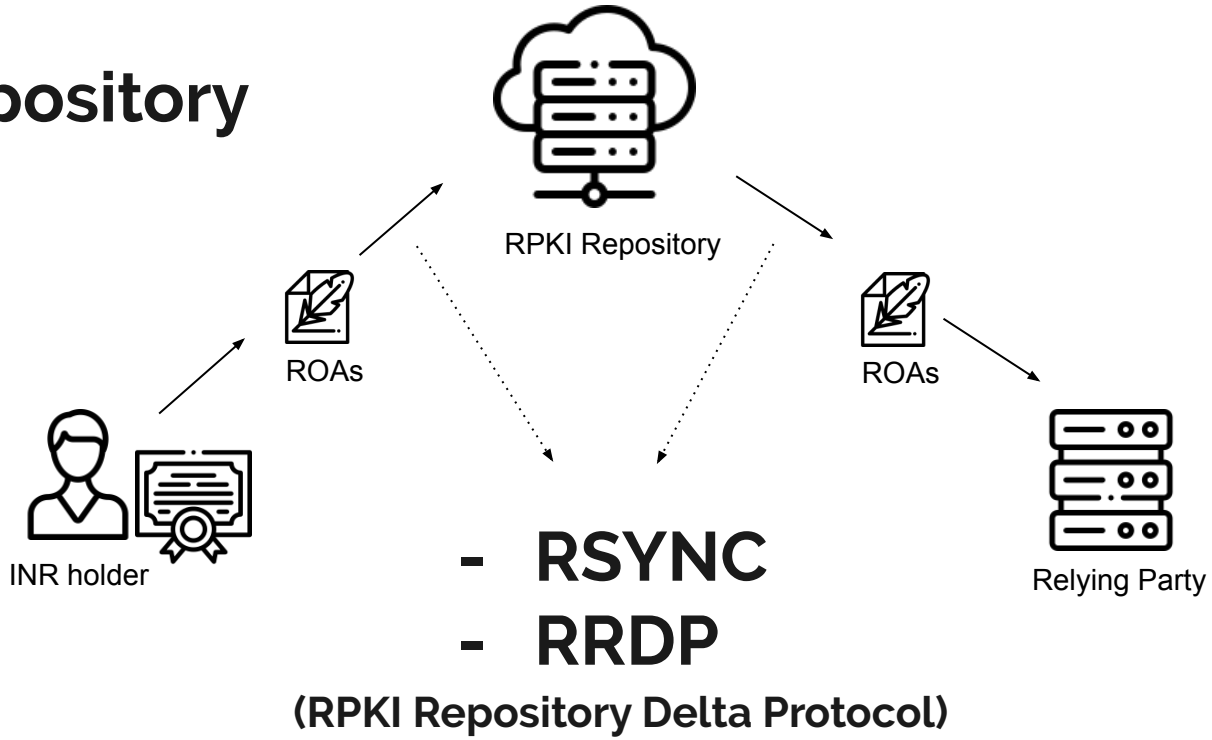


Fig. 4 - Publication components of RPKI

RSYNC Drawbacks

- Compute intensive.
- Lack of implementation library
- Atomic updates not guaranteed

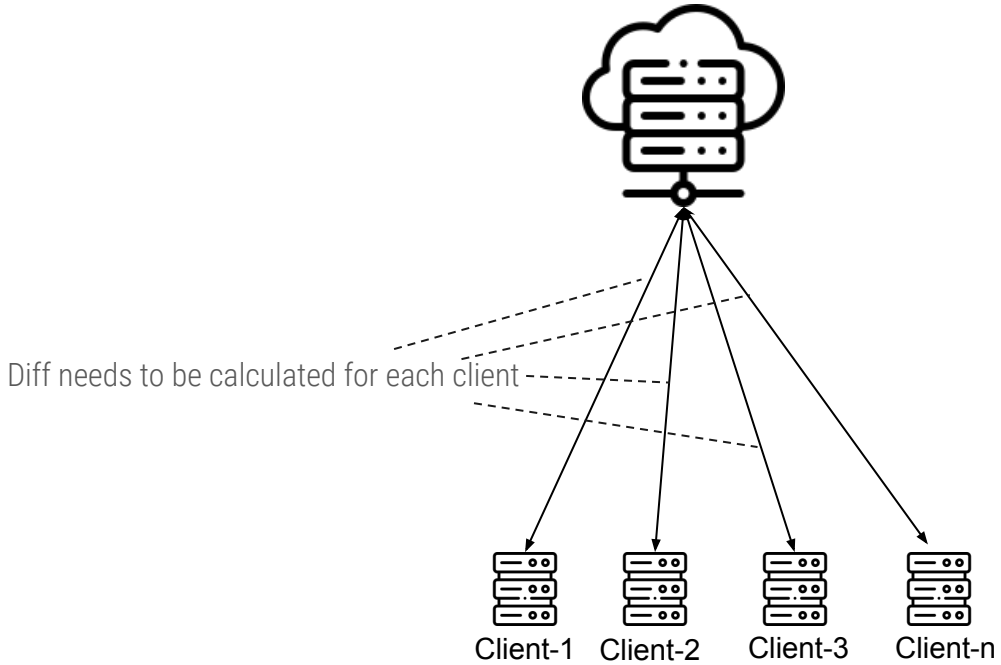


Fig. 5 -RSYNC server and clients

RRDP Improvements

- Reduces computation resources by generating Delta files once and not at every request
- Guarantees atomic updates
- Takes advantage of CDN and Caching Infrastructure.
- Uses HTTPS which has both client and server library implementations

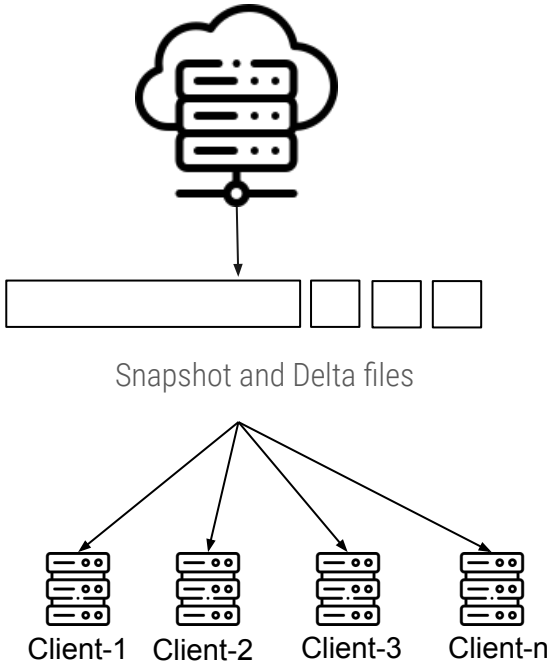


Fig.7 - HTTP server and clients using RRDP

Further Improvements Possible?

Research Question

To what extent can IPFS be used as a distribution mechanism within RPKI?

- How is publishing and retrieving contents currently implemented with RRDP in RPKI?
- What are the features of IPFS that can replace or augment the current RRDP implementation of the RPKI repository?
- What are the network characteristics of IPFS and how would these characteristics influence the operations of an RPKI repository?

Related Work

- No RRDP specific research to the best of our knowledge (Introduced in 2017).

- J. Shen et al. "**Understanding I/O Performance of IPFS Storage: A Client's Perspective**". In: 2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS). 2019, pp. 1-10.
- Netflix: **New improvements to IPFS Bitswap for faster container image distribution.**
- V. Kotlyar et al. "**Torrent Base of Software Distribution by ALICE at RDIG**". In: (2012), pp. 171-175.
- B. Confais, A. Lebre, and B. Parrein. **An Object Store Service for a Fog/Edge Computing Infrastructure Based on IPFS and a Scale-Out NAS**". In: 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC). 2017, pp. 41{50.
- IPFS for Off Chain Storage:
    - Sihua Wu and Jiang Du. **Electronic medical record security sharing model based on blockchain**".
    - R. Norvill et al. **IPFS for Reduction of Chain Size in Ethereum**".
    - Q. Zheng et al. **An Innovative IPFS-Based Storage Model for Blockchain**". In: 2018

Methodology - assessing network performance

- **Qualitative Analysis** - Literature study of RPKI, RRDP and IPFS[1,2]
- **Quantitative Analysis** - Direct HTTPs and IPFS comparison (exclude RSYNC to limit scope)
  - Compare data transfer
  - Test environment based on Containernet (Mininet) [2]
- **Quantitative Analysis** - HTTPs and IPFS comparison within RPKI (exclude RSYNC to limit scope)
  - Compare fetching of VRP
  - Modify Krill - RPKI Certificate Authority/Repository - to use IPFS] [3]
  - Modify Routinator - RPKI Relying Party software - to use IPFS [4]
  - Test environment based on Docker containers using Docker Compose [5]

1 **References:** RFC 8630, RFC 8182, RFC 6486, RFC 6482, RFC 6480, IPFS-Specification, IPFS Documentation, 2 More available in the report

Results

- **Qualitative Analysis**
  - Removing the need for hashes in notification.xml
- **Quantitative Analysis** - Direct HTTPs and IPFS comparison (exclude RSYNC to limit scope)
  - Bandwidth test
- **Quantitative Analysis** - HTTPs and IPFS comparison within RPKI (exclude RSYNC to limit scope)
  - Number of nodes test

Remove checksum in RRDP notification file

IPFS uses content addressing, hence cryptographic hash of contents can be used for both retrieval and assurance of integrity

- **Current notification.xml**

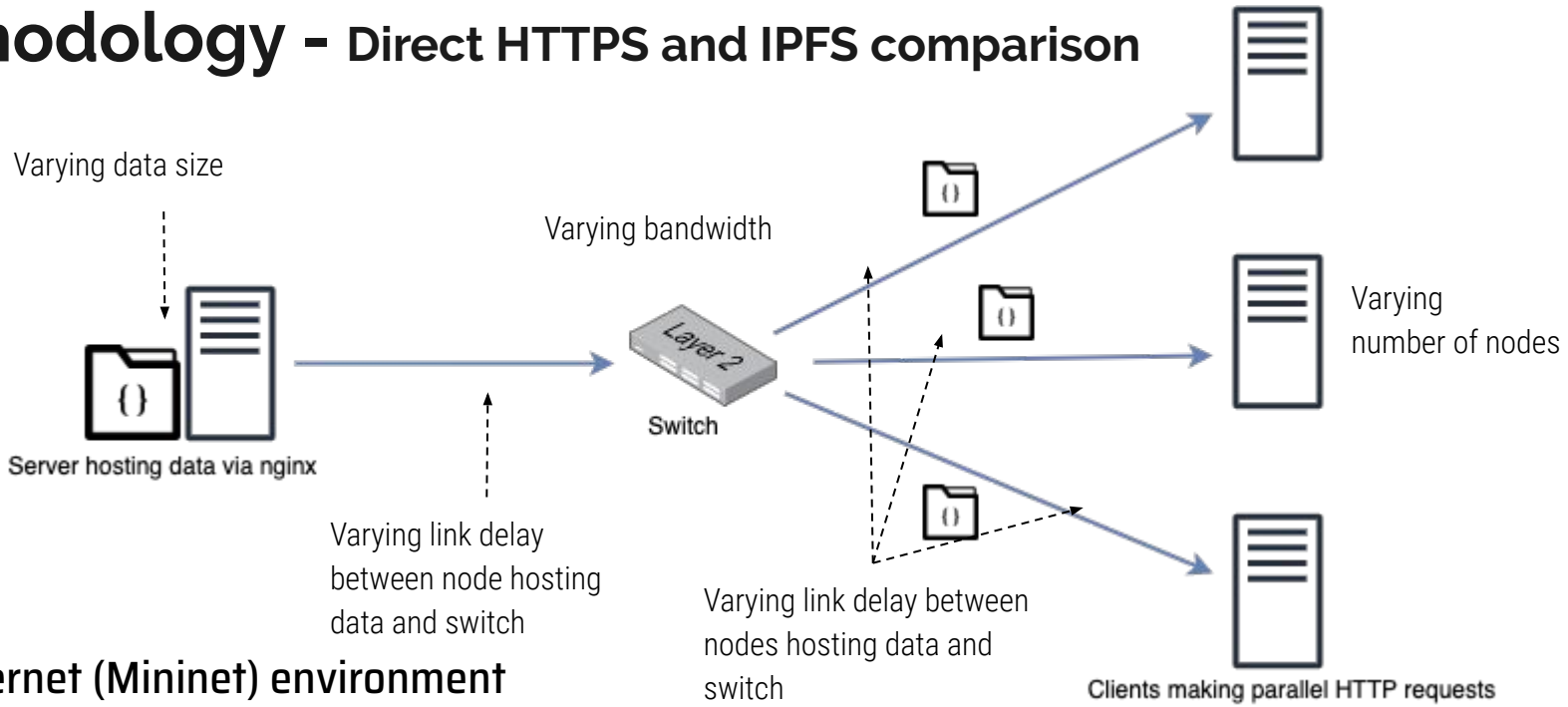
```
<notification xmlns="http://www.ripe.net/rpki/rrdp" version="1" session_id="56a98049-6402-4b58-ac6f-c3c395293498" serial="776">
    <snapshot uri="https://url/snapshot.xml" hash="83420CD0F19533DC368C485DC4EC2D07D48D9AFB8C42ECFEA0E1C2FABFA284DE"/>
    <delta serial="776" uri="https://url/delta.xml" hash="AC7DE0CEE5836F51240E01055473FAD9AA78B3E971211AB6DC7C1A1FAF67927F"/>
</notification>
```

- **Possible modification using IPFS**

```
<notification xmlns="http://www.ripe.net/rpki/rrdp" version="1" session_id="56a98049-6402-4b58-ac6f-c3c395293498" serial="776">
    <snapshot cid="83420CD0F19533DC368C485DC4EC2D07D48D9AFB8C42ECFEA0E1C2FABFA284DE"/>
    <delta serial="776" cid="AC7DE0CEE5836F51240E01055473FAD9AA78B3E971211AB6DC7C1A1FAF67927F" />
</notification>
```

Fig.8 - RRDP notification.xml file without and with IPFS based modification

Methodology - Direct HTTPS and IPFS comparison



Containernet (Mininet) environment

Fig.9 - Network topology for direct HTTPs and IPFS comparison

Results - IPFS latency test

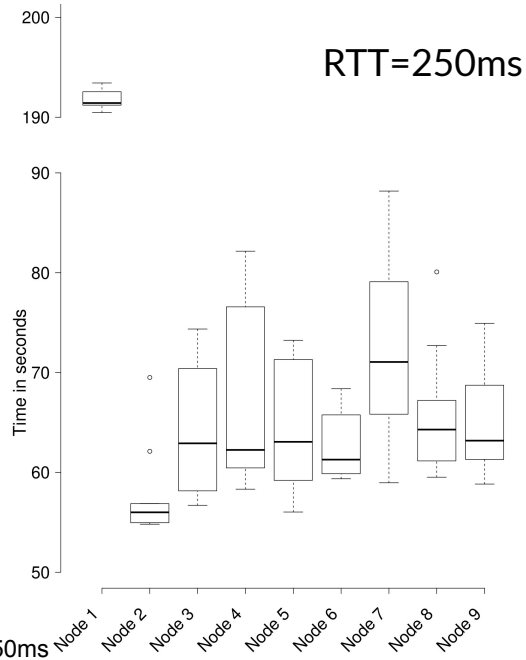


Fig.10 - IPFS w/ RTT of 250ms

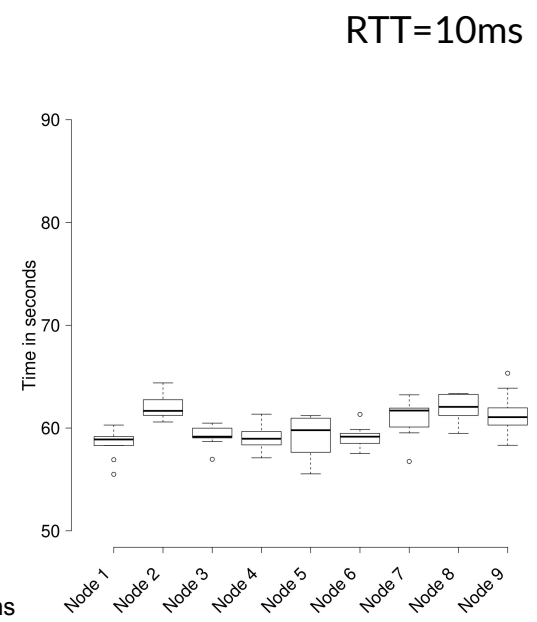


Fig.11 - IPFS w/ RTT of 10ms

Results - HTTPs latency test

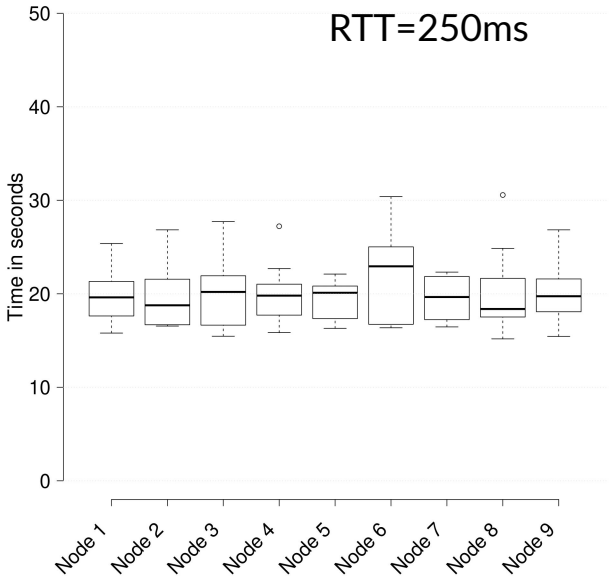


Fig.12 - HTTPS w/ RTT of 250ms

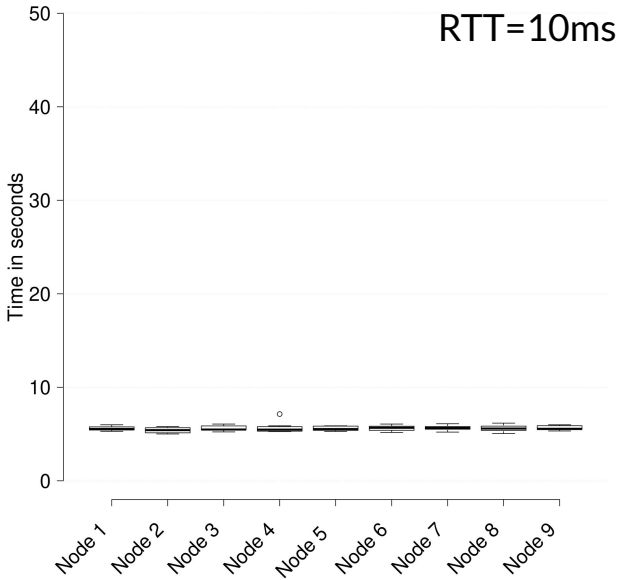
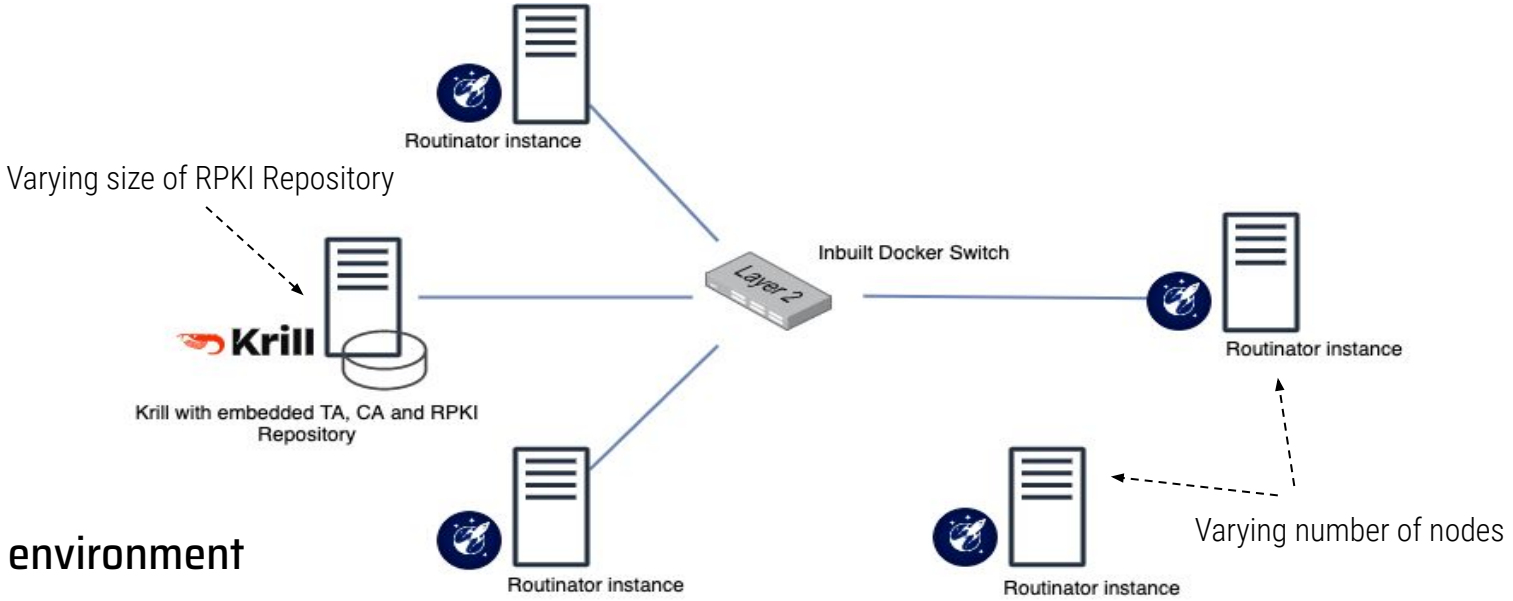


Fig.13 - HTTPS w/ RTT of 10ms

Methodology - RRDP and IPFS comparison within RPKI



Docker environment

Fig.14 - Network topology for HTTPs and IPFS comparison within RPKI

Results - RPKI IPFS nodes test

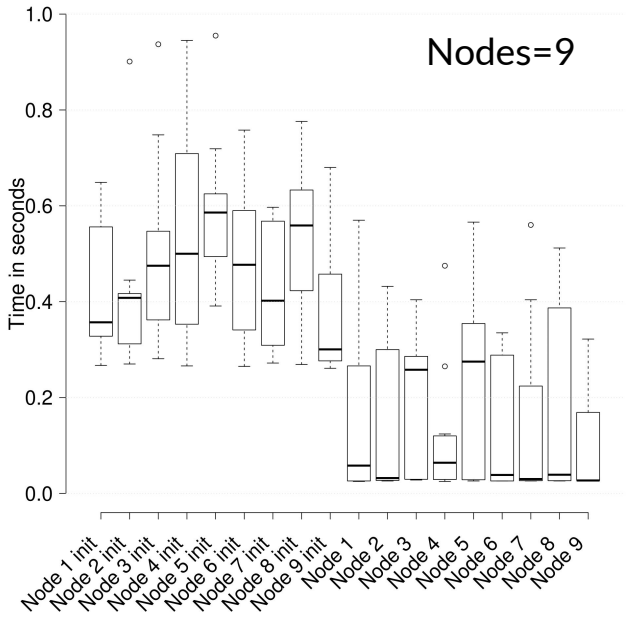


Fig.15 - RPKI IPFS w/ 9 nodes

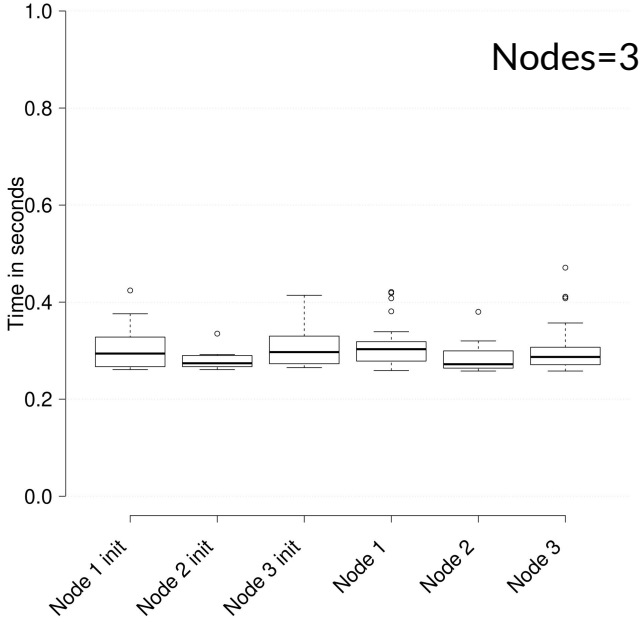


Fig.16 - RPKI IPFS w/ 3 nodes

RPKI RRDP nodes test

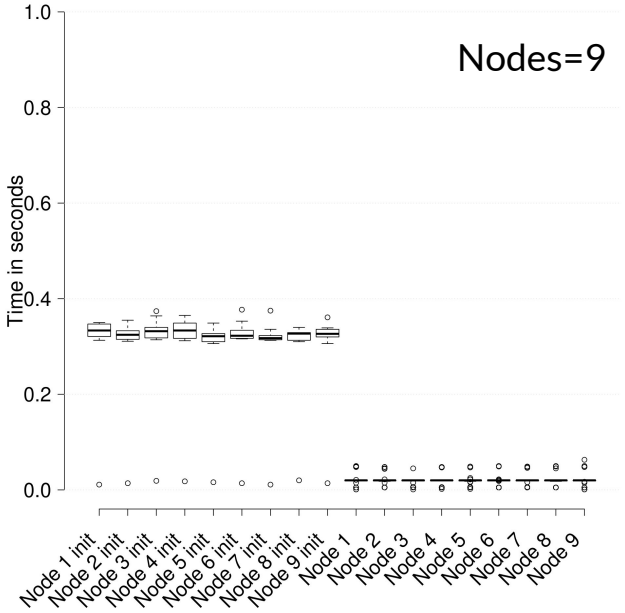


Fig.17 - RPKI RRDP w/ 9 nodes

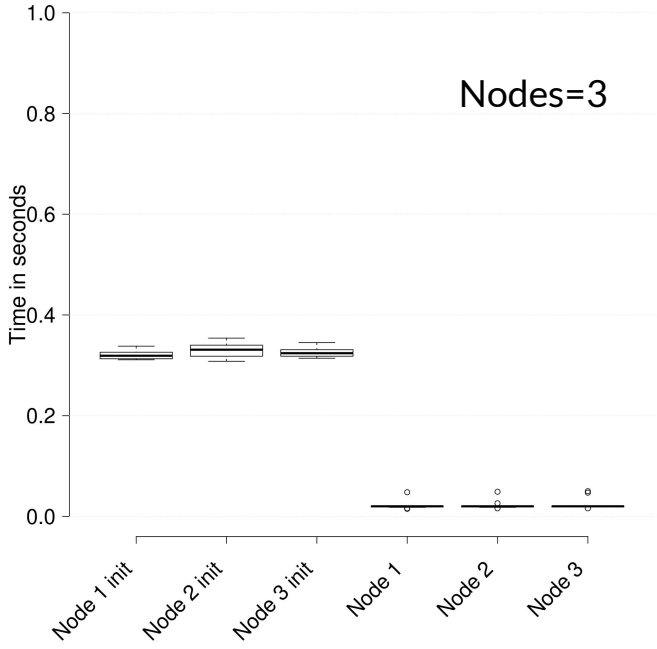


Fig.18 - RPKI RRDP w/ 3 nodes

Conclusion

- IPFS can currently be integrated to distribute RPKI material
  - Removing the need for manual data integrity checks in RRDP
- IPFS performed poorly in direct comparison with HTTPS:
  - Retrieval times were several factors higher than HTTPs under the same circumstances
  - In the low bandwidth, low latency environment it only performed 1.5x as poorly

	Low latency(RTT=10ms)	High latency(RTT=250ms)
Low bandwidth(100Mbit/s)	HTTPs	N/a
High bandwidth(1000Mbit/s)	HTTPs	HTTPs

Future Work

- Research variable delays between retrieving IPFS nodes, not only the server hosting the data
- Research effect of concurrent requests in IPFS (without RPKI)
- Research power consumption of IPFS in comparison to other transfer protocols
- Research integration of IPFS into Krill and Routinator using the IPFS Rust library[6] (once matured)

Thank you for your attention

In short:

- The network is often not the bottleneck in IPFS performance, it is more susceptible to I/O
- IPFS can be integrated into RPKI and replace redundant functionality

References

1. Baran, P. (1962). On Distributed Communications Networks. RAND Corporation. Setembro de
2. sne-os3-rp2/ipfshttpbenchmark : Containernet script for performing data transfer benchmark of HTTPs and IPFS:. url: https : / / github . com / sne - os3 - rp2 / ipfs _ http _ benchmark (visited on 06/27/2020).
3. sne-os3-rp2/krill: RPKI Certicate Authority and Publication Server written in Rust. url: https://github.com/sne-os3-rp2/krill (visited on 06/28/2020).
4. sne-os3-rp2/routinator: An RPKI Validator written in Rust. url: https://github.com/sne-os3-rp2/routinator (visited on 06/28/2020).
5. sne-os3-rp2/lab: Scripts, and Docker build les for creating Docker compose le that is to be used to orchestrate Krill and routinator instances for experiments purposes. url: https://github.com/sne-os3-rp2/lab (visited on 06/28/2020).
6. rs-ipfs/rust-ipfs: The Interplanetary File System (IPFS), implemented in Rust.url:https://github.com/rs-ipfs/rust-ipfs(visited on 07/01/2020)
7. Benet, Juan. (2014). IPFS - Content Addressed, Versioned, P2P File System.