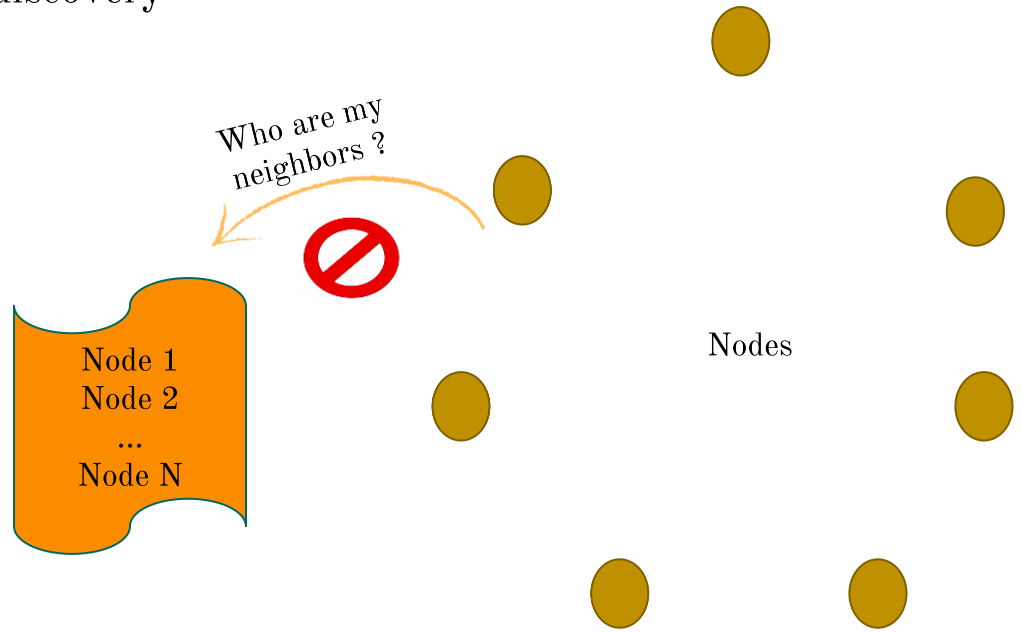
AUPE: Collaborative Byzantine fault-tolerant peer-sampling

NCA'24

Augusta Mukam, Joachim Bruneau-Queyreix, Laurent Réveillère

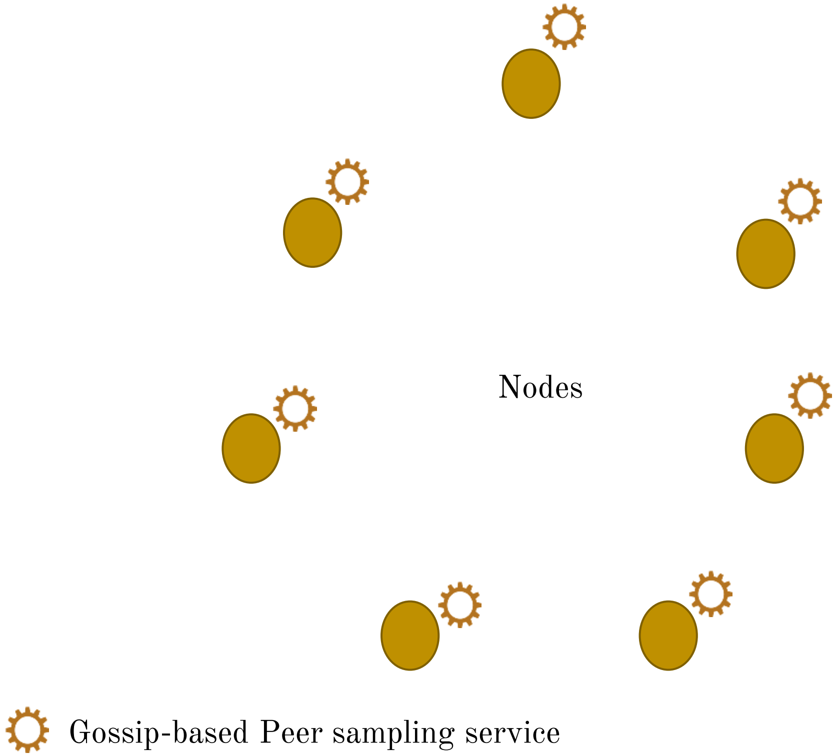
Large scale distributed systems

- No central tracking for peer discovery



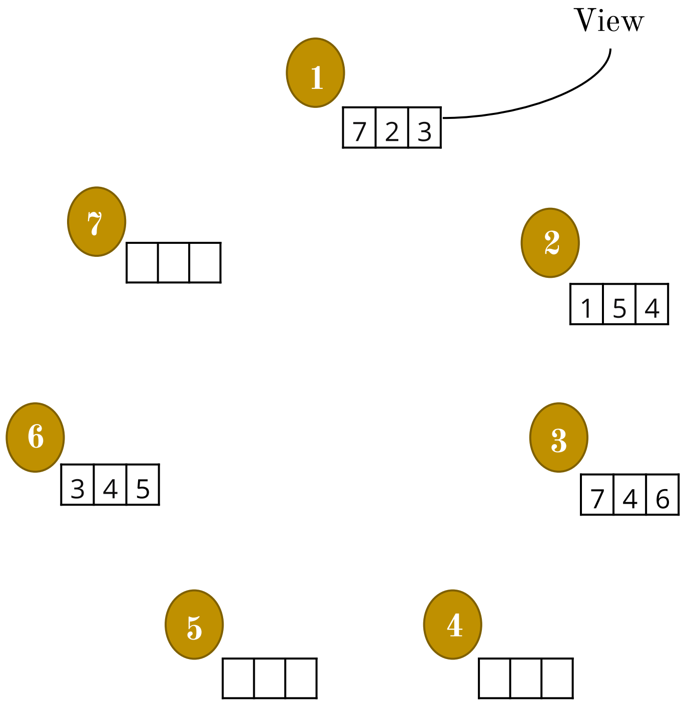
Large scale distributed systems

- No central tracking for peer discovery
- **Gossip-based peer sampling**
  - Aim: Maintain knowledge of active nodes
  - For selecting and providing random & uniform samples of identifiers (IDs)



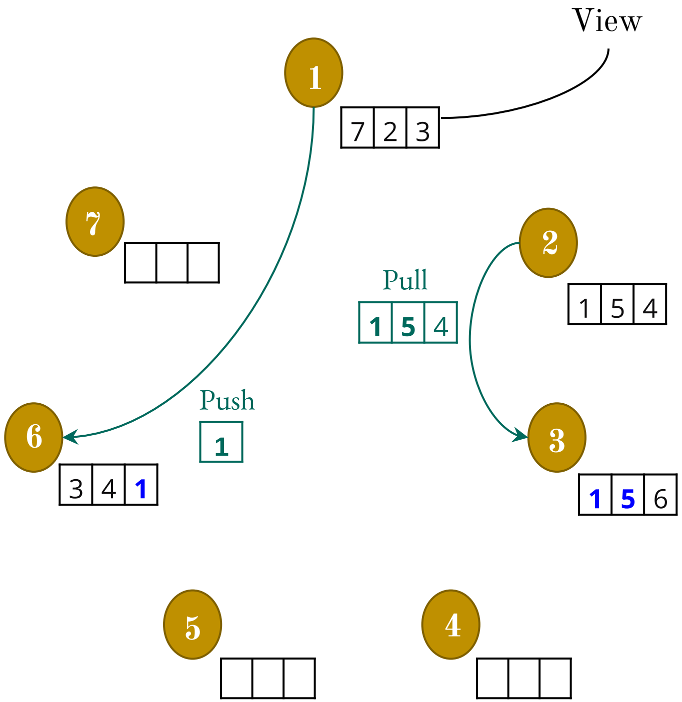
Gossip-based peer sampling service

- Each node has a local **View**



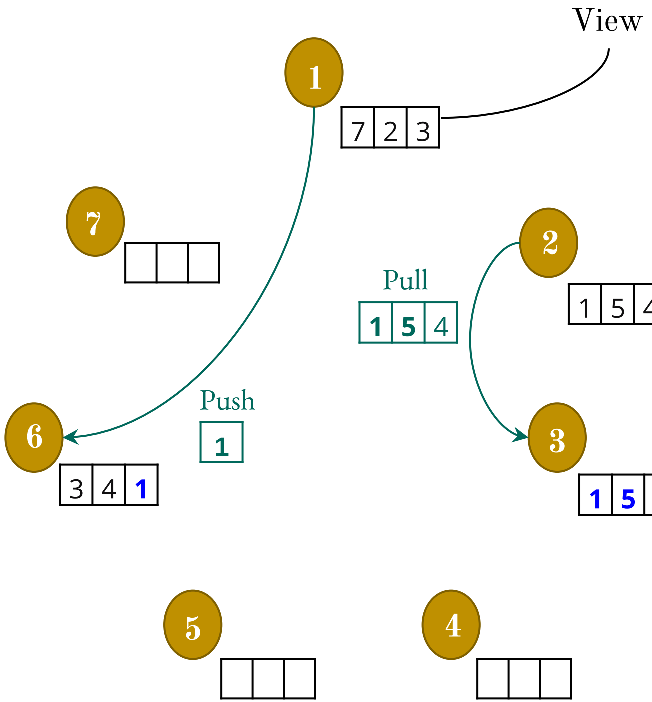
Gossip-based peer sampling service

- Each node has a local **View**
- Periodically:
  - Exchange **Push** and **Pull** requests
  - Update view



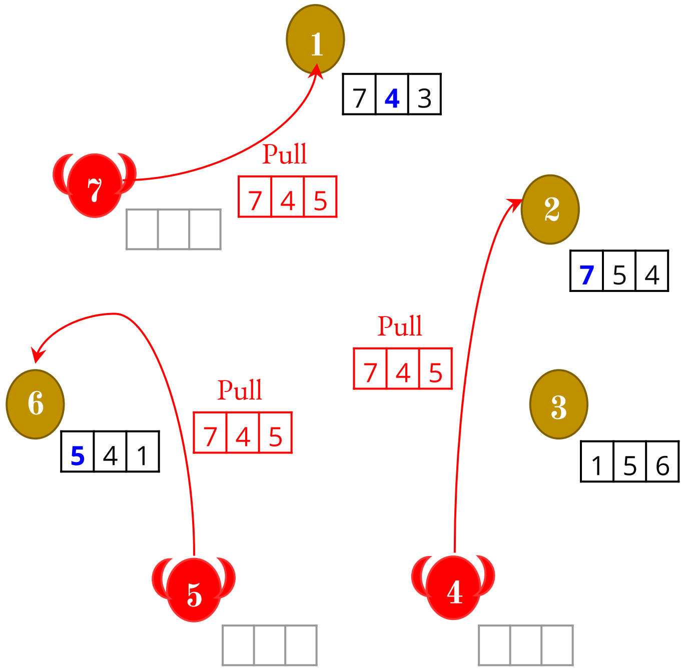
Gossip-based peer sampling service

- Each node has a local **View**
- Periodically:
  - Exchange **Push** and **Pull** requests
  - Update view
- Global network connectivity



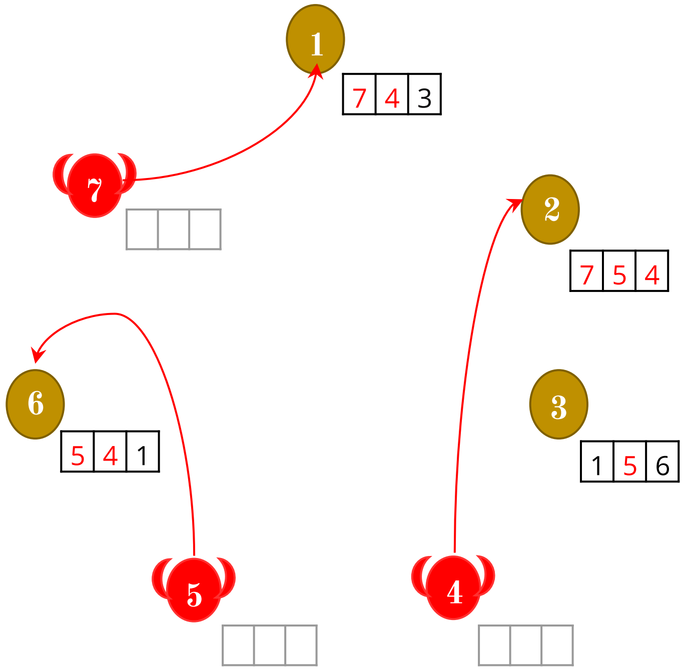
Problem

- Group of malicious/Byzantine nodes
- Promote nodes within their member group



Problem

- Group of malicious/Byzantine nodes
- Promote nodes within their member group
- Increase their representation in honest nodes views

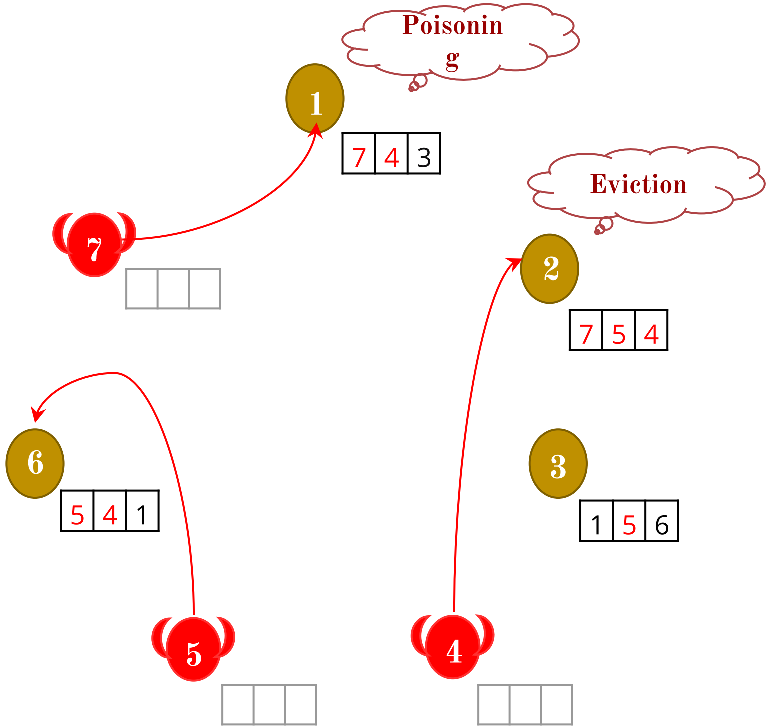


Problem

- Group of malicious/Byzantine nodes
- Promote nodes within their member group
- Increase their representation in honest nodes views

Eg: Bitcoin Eclipse attack

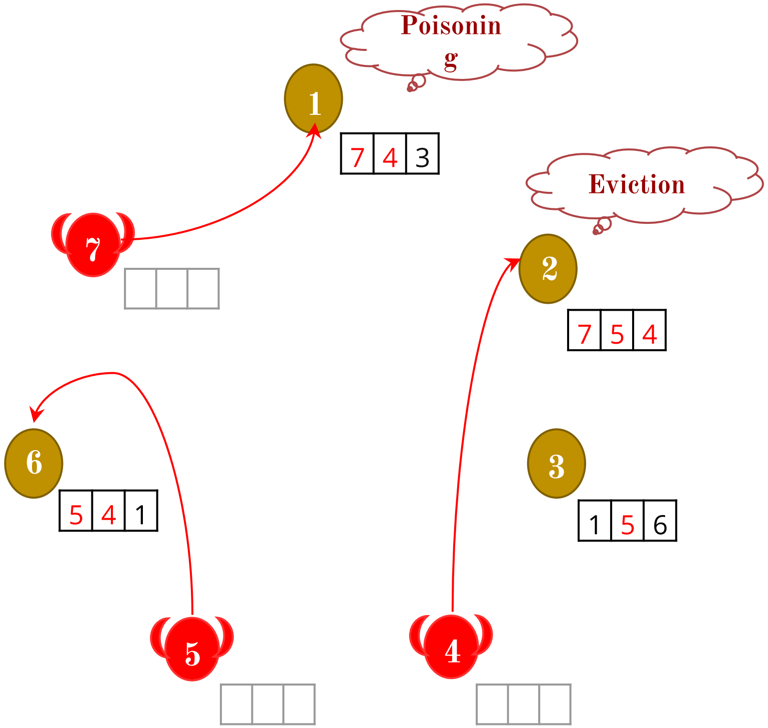
- Corrupted version of blockchain
- Manipulate their tokens



Problem

- Group of malicious/Byzantine nodes
- Promote nodes within their member group
- Increase their representation in honest nodes views

- Target higher-level protocols
- Network Attacks



Fault-detection

- Identify malicious nodes based on misbehavior proofs
- Punish faulty nodes
- Lead to major disruption

Fault-tolerance

- Tolerate malicious nodes
- Prevent them from polluting the system

Fault-tolerance

- Tolerate malicious nodes
- Prevent them from polluting the system
- **Brahms,** extension **Basalt**

Fault-tolerance

- Tolerate malicious nodes
- Prevent them from polluting the system
- **Brahms,** extension **Basalt**

Brahms

f=26% malicious nodes

$\downarrow$

77% malicious IDs in honest node views

Fault-tolerance

- Tolerate malicious nodes
- Prevent them from polluting the system
- **Brahms,** extension **Basalt**

Brahms

$f=26\%$ malicious nodes

↓

77% malicious IDs in honest node views

Basalt

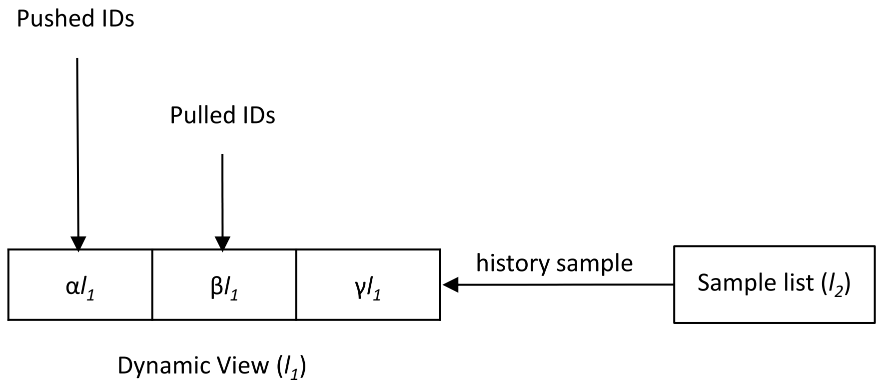
Better than Brahms for $\mathbf{f< 20\%}$

Results get worse rapidly

BRAHMS overview

Gossip component

- Handle push/pull requests
- View update



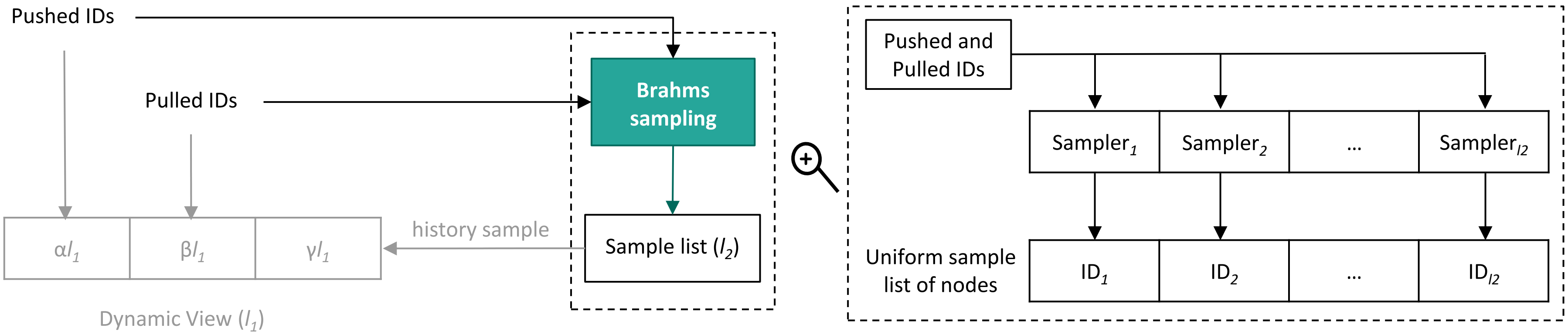
BRAHMS overview

Gossip component

- Share knowledge
- View update

Sampling component

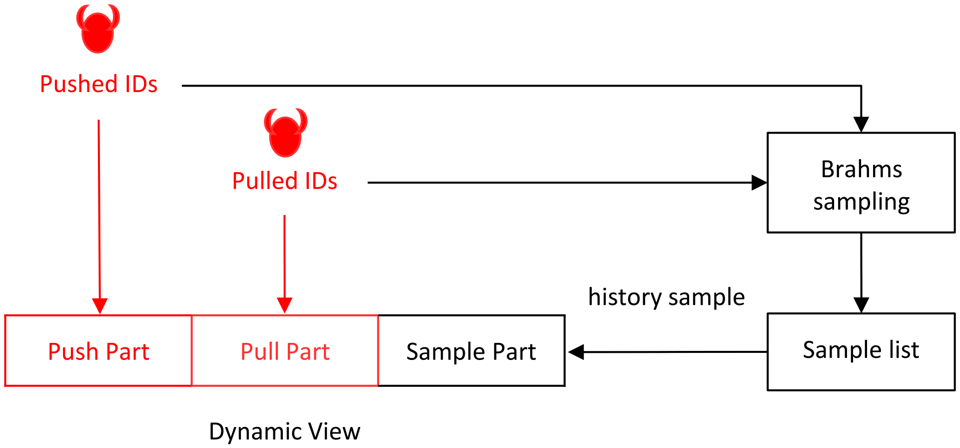
- Uniform sample of seen nodes



Motivation

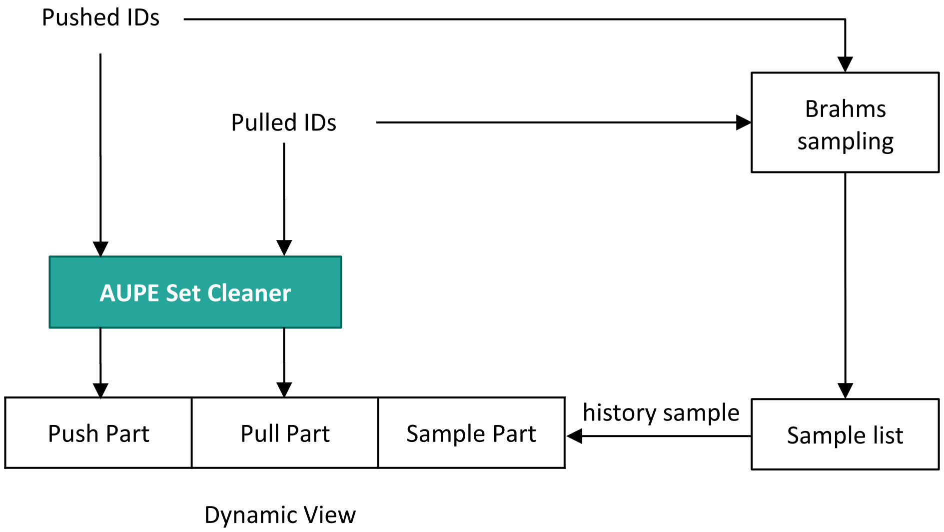
➔ Received streams of identifiers are source of bias

➔ **Mitigate Byzantine over representation**



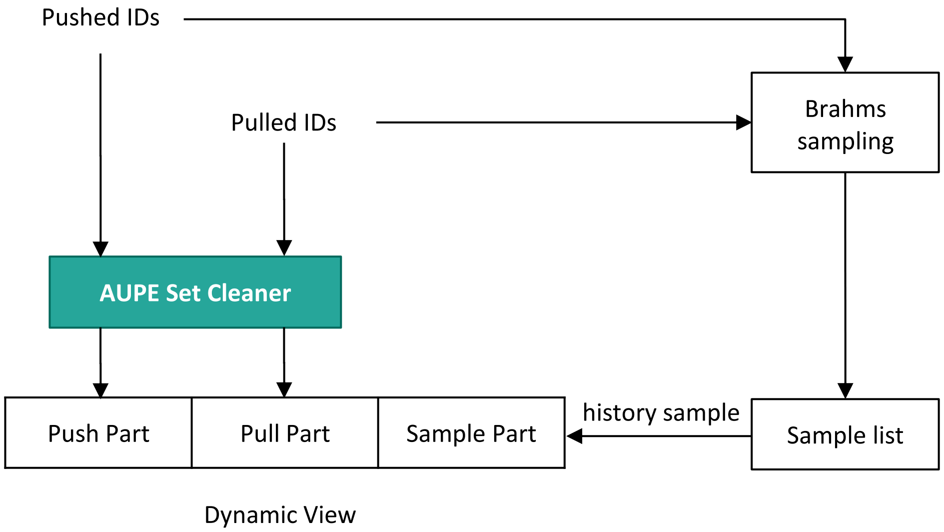
AUPE Protocol

- Based on BRAHMS components
- **AUPE Set Cleaner**
    - Produces less biased streams



AUPE Protocol

- Based on BRAHMS components
- **AUPE Set Cleaner** 
  - Produces less biased streams
- **AUPE Secret Collaborative debiasing** 
  - Enhance the local debiasing mechanism



AUPE Set Cleaner

Tracking component

- Record occurrences of received IDs in a tracking data-structure

AUPE Set Cleaner

Tracking component

- Record occurrences of received IDs in a tracking data-structure

Tracking data-structure

Key-value store

- Give real occurrences
- Same size as the system

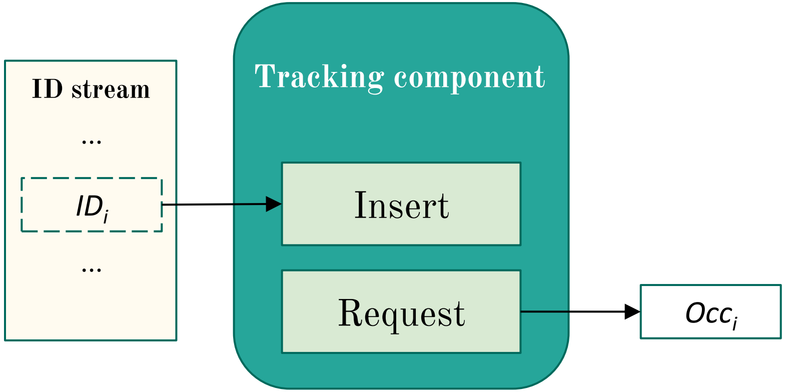
Count-min-sketches

- Probabilistic data-structure
- Give estimate occurrences
- Fixed-size

AUPE Set Cleaner

Tracking component

- Record occurrences of received IDs in a tracking data-structure



Occurrence of node i (real or estimated):
Occ_i

AUPE Set Cleaner

Debiasing component

- Transforms received stream to a more uniform one
- Probability of inserting into **sample memory**

Probability of insertion of ID i : p_i
Minimum of all occurrences : min

$$p_i = \frac{min}{Occ_i}$$

AUPE Set Cleaner 

Debiasing component

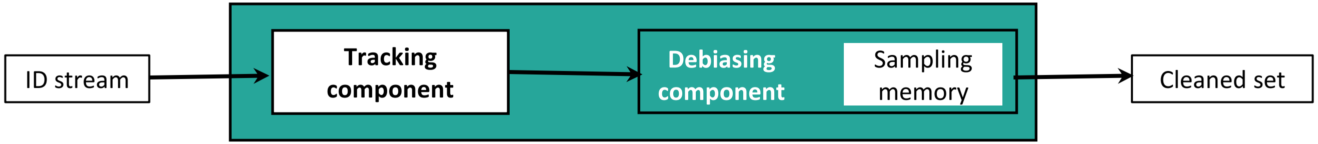
- Transforms received stream to a more uniform one
- Probability of inserting into **sample memory**
- Sample memory IDs form the output stream

Probability of insertion of ID i : p_i
Minimum of all occurrences : min

$$p_i = \frac{min}{Occ_i}$$

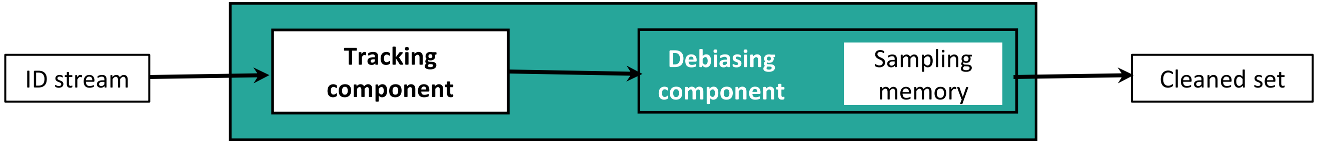
AUPE Set Cleaner review

- Choose infrequent IDs more often
- Improve correct node tolerance to malicious over-representation



AUPE Set Cleaner review

- Choose infrequent IDs more often
- Improve correct node tolerance to malicious over-representation



Increase of Brahms tolerance by up to 60%

AUPE Secret Collaborative Debiasing 🤝

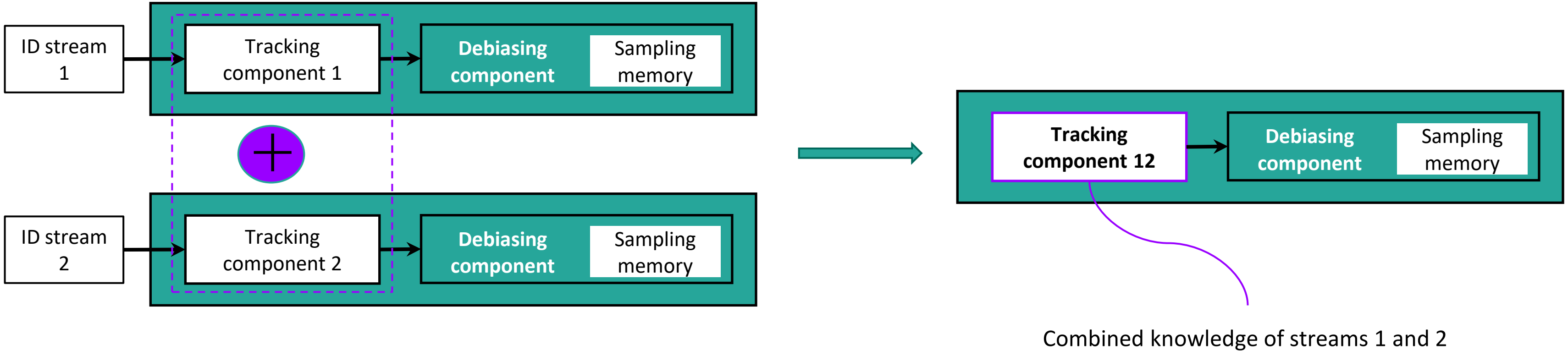
- System is equipped with **Trusted nodes**
  - Based on TEE technology: authenticity of the code
  - **Secure mutual authentication** to recognize trusted peers

AUPE Secret Collaborative Debiasing

- System is equipped with **Trusted nodes**
  - Based on TEE technology: authenticity of the code
  - **Secure mutual authentication** to recognize trusted peers
- **Exchange** and **merge** their tracking components
- Enhance the debiasing mechanism of the Set Cleaner

AUPE Secret Collaborative Debiasing

- **Merge** ⊕ : Aggregate two tracking components
  - **Average** computation of each corresponding entries



AUPE Secret Collaborative Debiasing

- **Merge** : Aggregate two tracking components
  - **Average** computation of each corresponding entries
- **Trusted peer list**
  - M last known trusted peer IDs to recontact

Evaluation questions

- To what extent does **Aupe-simple** (without Merge) improve the tolerance ?
- What is the impact of the **secret collaborative debiasing** ?
- Compare to **Brahms**, **Basalt**

Experimental evaluation

Metric

- **Resilience:** proportion of Byzantine IDs in honest node views at last round
- **Optimal Case:** system resilience is equal to system proportion of Byzantine nodes

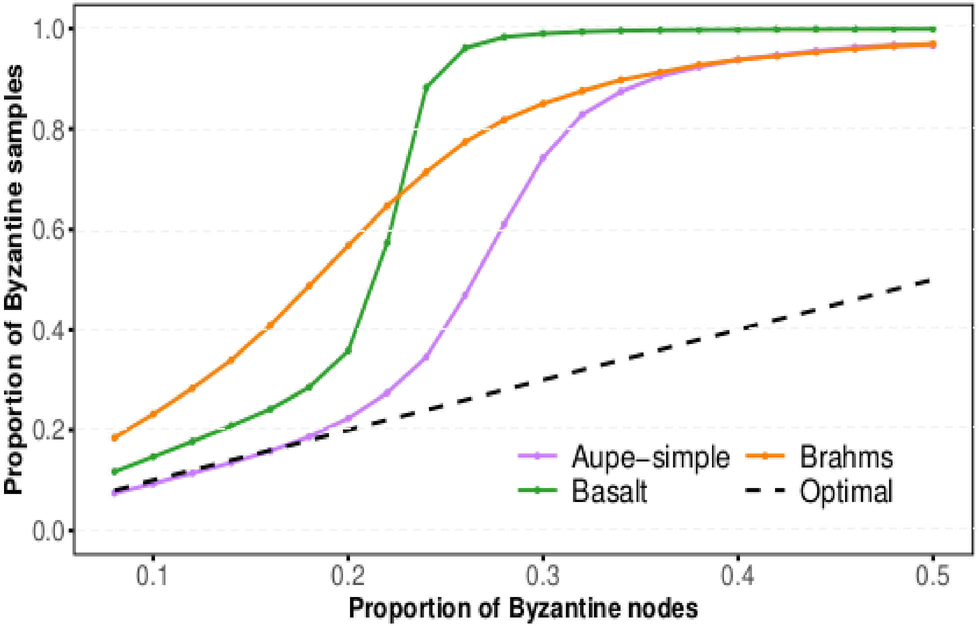
Experimental evaluation

Parameters

- System size **N**=10,000
- Fraction **f** of faulty nodes
- Fraction **t** of trusted nodes
- Tracking component : Key-value store

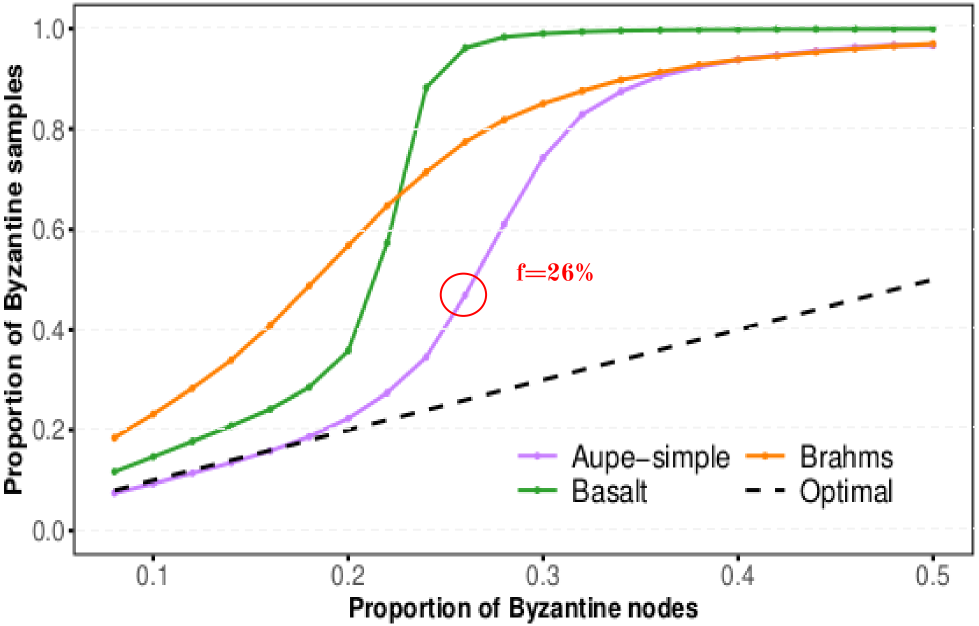
System Tolerance improvement

Aupe-simple



System Tolerance improvement

Aupe-simple

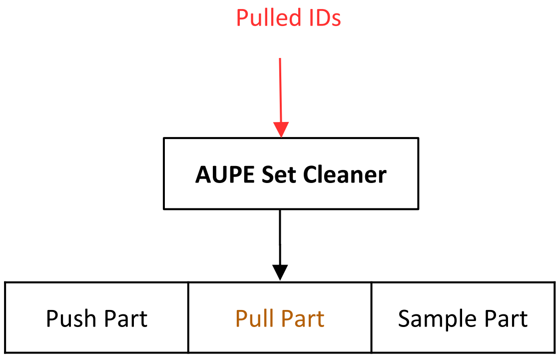


f=26%

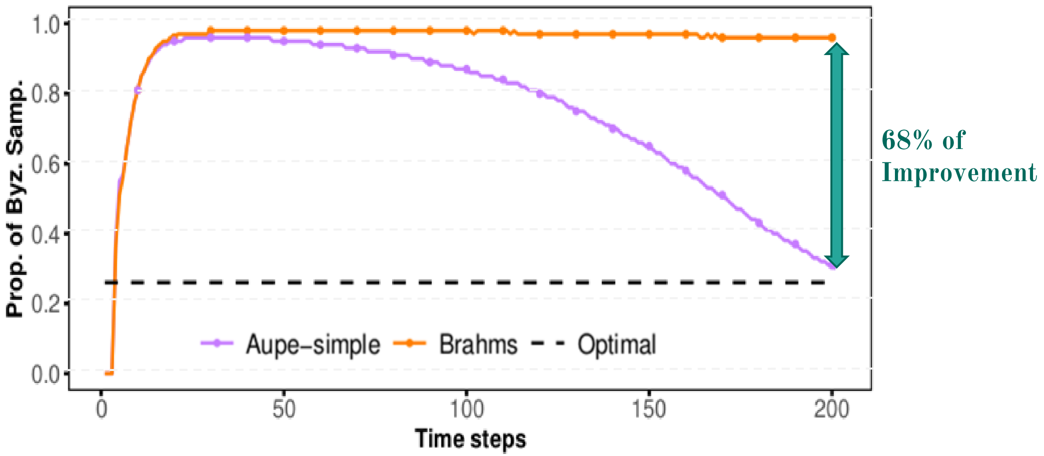
- ➔ **40%** on Brahms
- ➔ **52%** on Basalt

View parts tolerance improvement

Aupe-simple



f=26%

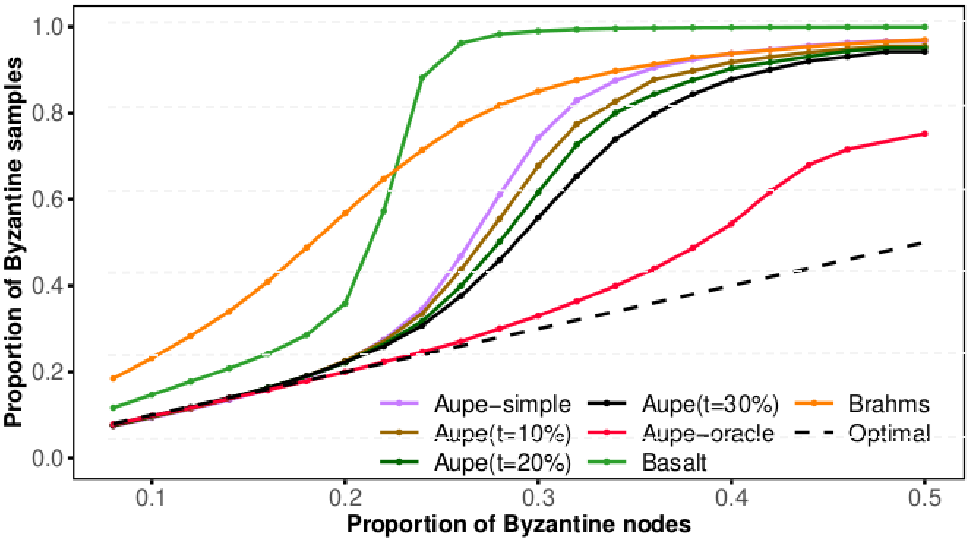


View' Pull part

Collaborative debiasing

Aupe with t=10%, 20% and 30%

- Good impact of collaborative debiasing



Conclusion


- **AUPE**
  - The first peer sampling that utilizes **Collaborative trusted debiasing** to achieve Byzantine-tolerance
- Near-perfect resilience
  - Even with adversary controlling **26%** of nodes

Conclusion


- **AUPE**
  - The first peer sampling that utilizes **Collaborative trusted debiasing** to achieve Byzantine-tolerance
- Near-perfect resilience
  - Even with adversary controlling **26%** of nodes
- Study trusted node re-identification attack

AUPE: Collaborative Byzantine fault-tolerant peer-sampling

NCA’24

Augusta Mukam, Joachim Bruneau-Queyreix, Laurent Réveillère

References

- *E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer, "**Brahms:** Byzantine resilient random membership sampling," in Proceedings of the Twenty-Seventh ACM Symposium on Principles of Distributed Computing, ser. **PODC** '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 145–154.*
- *E. Anceaume, Y. Busnel, and B. Sericola, "**Uniform node sampling service** robust against collusions of malicious nodes," in 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**), 2013, pp. 1–12*
- *M. Jelasity, A. Montresor, and O. Babaoglu, "**Gossip-based aggregation in large dynamic networks**," ACM Trans. Comput. Syst., vol. 23, no. 3, p. 219–252, aug 2005.*
- *A. Auvolat, Y.-D. Bromberg, D. Frey, D. Mvondo, and F. Taïani, "**Basalt:** A rock-solid byzantine-tolerant peer sampling for very large decentralized networks," in Proceedings of the 24th International Middleware Conference, ser. Middleware '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 111–123.*