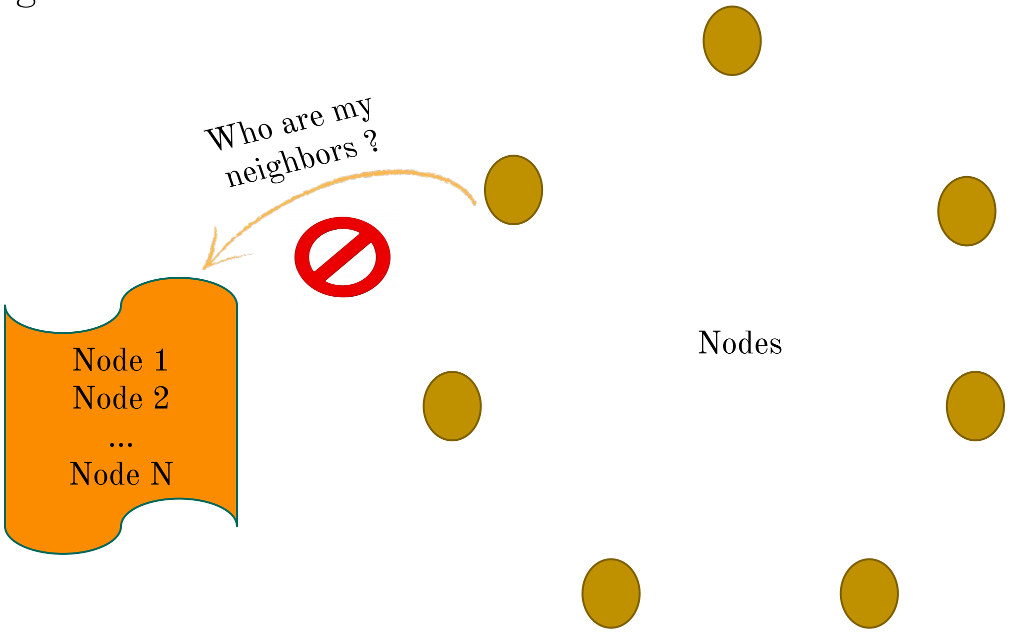
AUPE:
Collaborative Byzantine fault-tolerant peer-sampling

Compas'25

Augusta Mukam, Joachim Bruneau-Queyreix, Laurent Réveillère

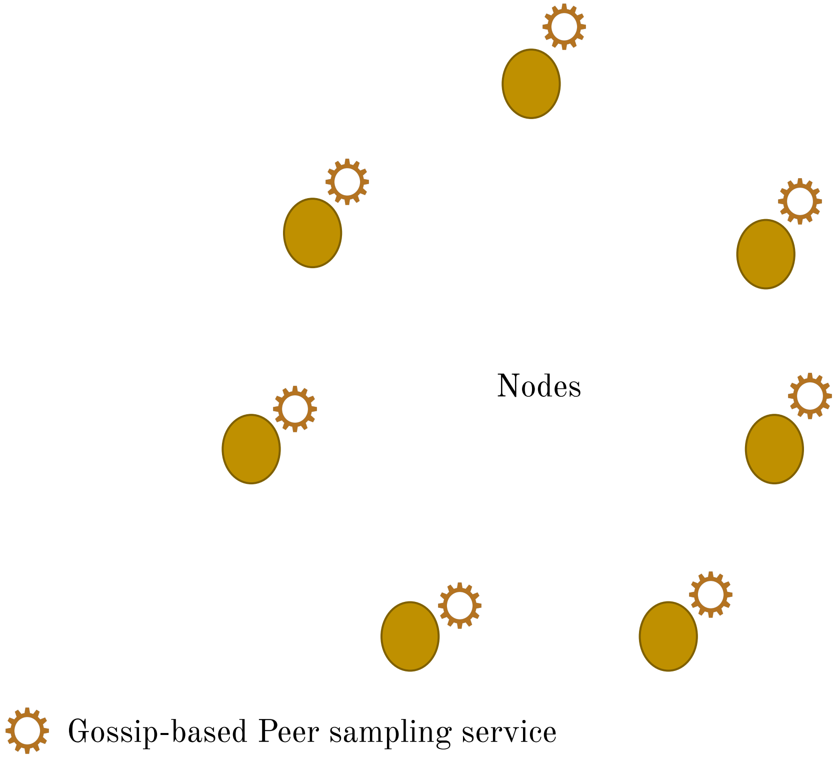
Large scale distributed systems

- No tracking component for neighbors listing



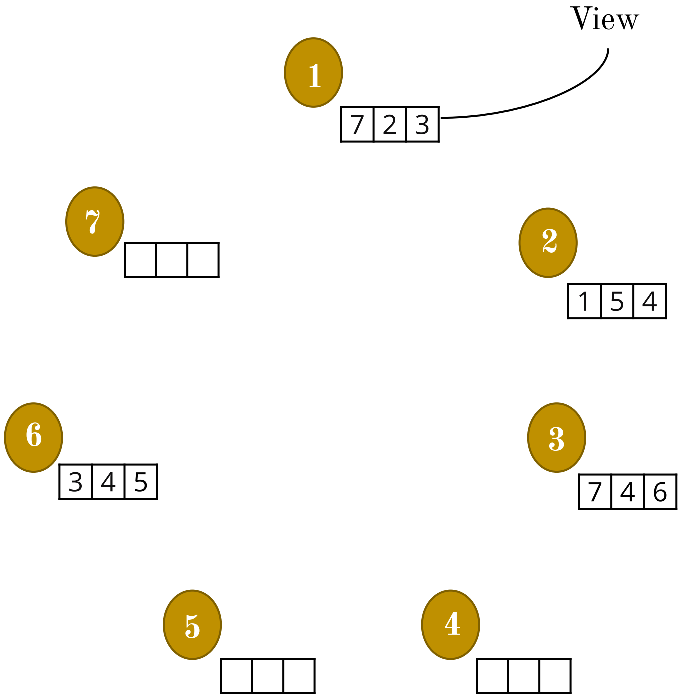
Large scale distributed systems

- **Gossip-based peer sampling**
  - Aim: Maintain knowledge of active nodes
  - For selecting and providing random & uniform samples of identifiers (IDs)



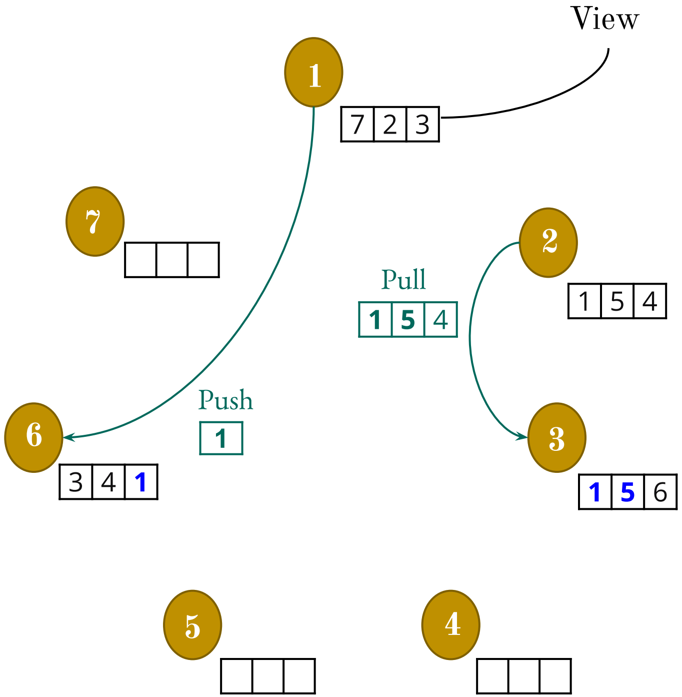
Gossip-based peer sampling service

- Each node has a local **View**



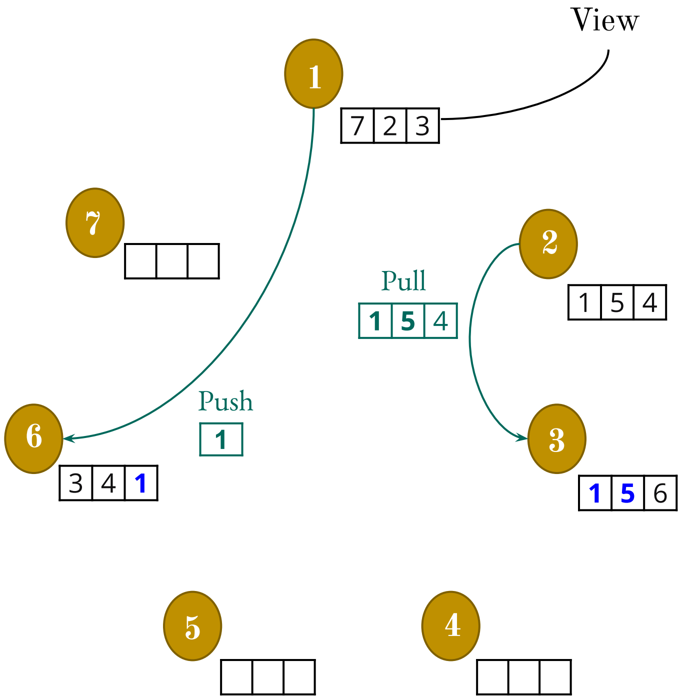
Gossip-based peer sampling service

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



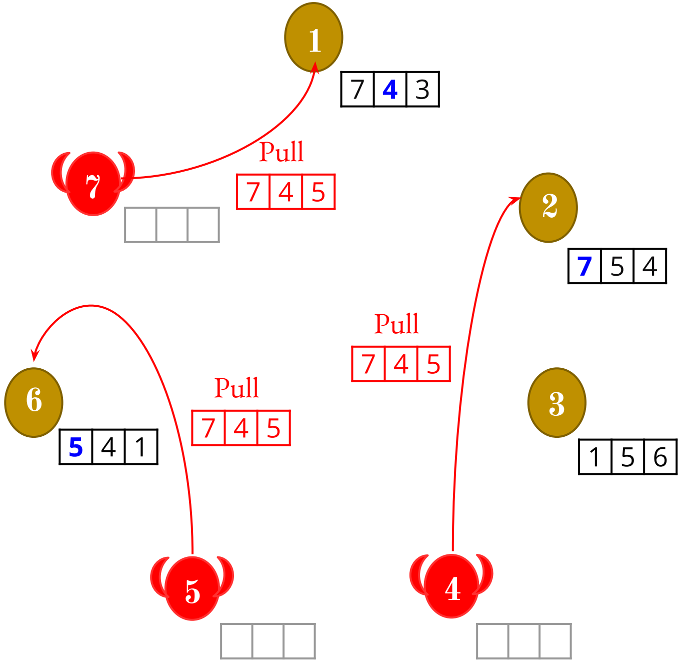
Gossip-based peer sampling service

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



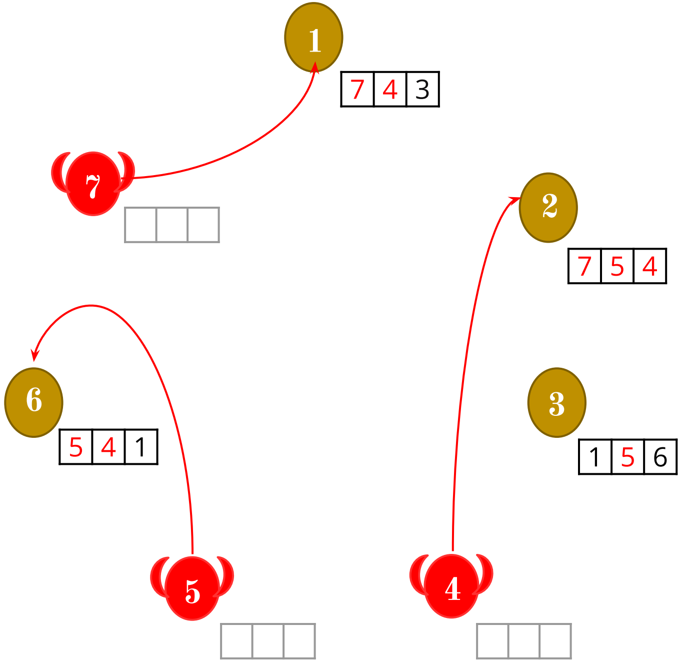
Problem

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



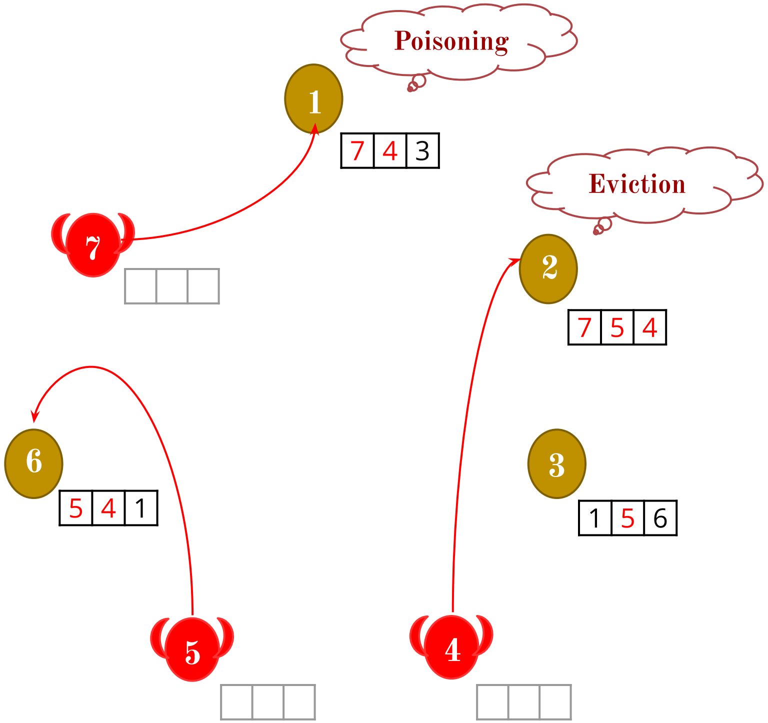
Problem

- 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



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

77% malicious IDs in honest node views

Basalt

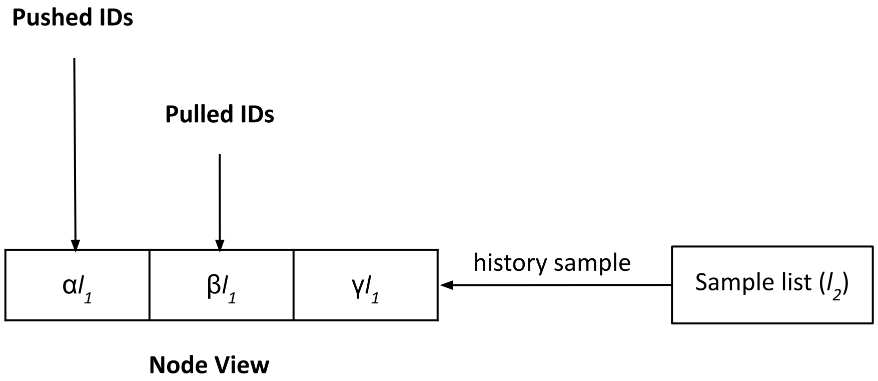
Better than Brahms for **$f< 20\%$**

Results get worse rapidly

BRAHMS overview

Gossip component

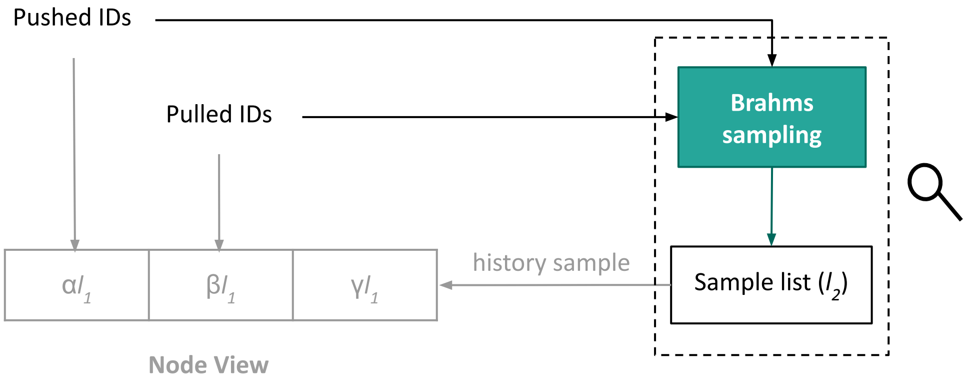
- Handle **push/pull** requests
- View update



BRAHMS overview

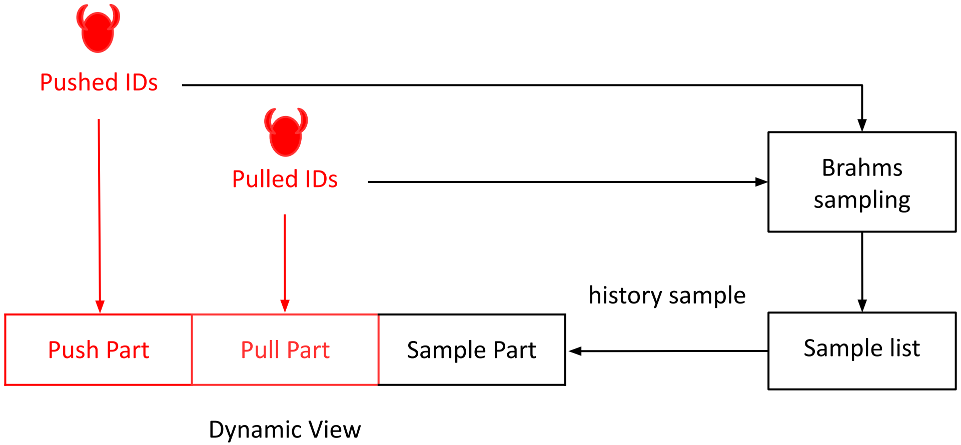
Sampling component

- Set of hash functions
- Uniform sample of seen nodes



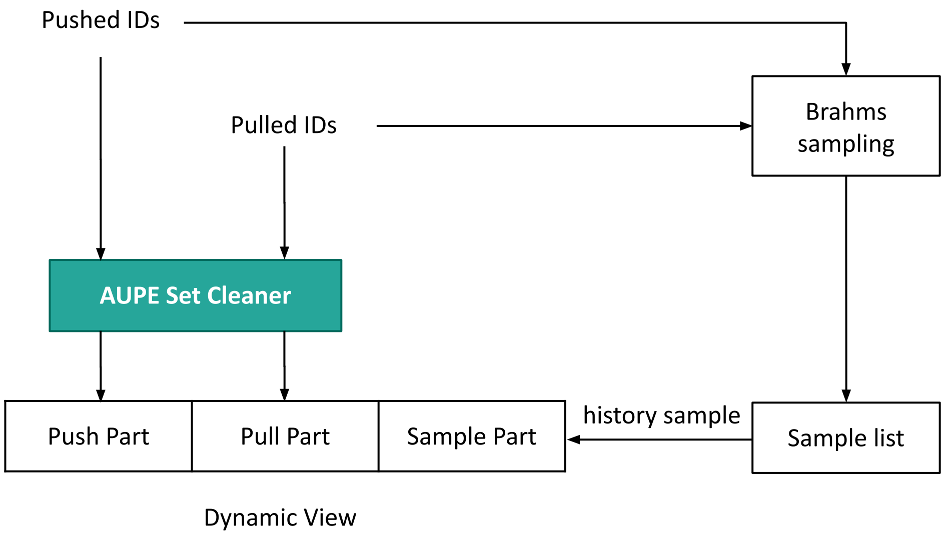
Motivation

- ➔ Received streams of identifiers are source of bias
- ➔ **Mitigate Byzantine over representation inside streams**



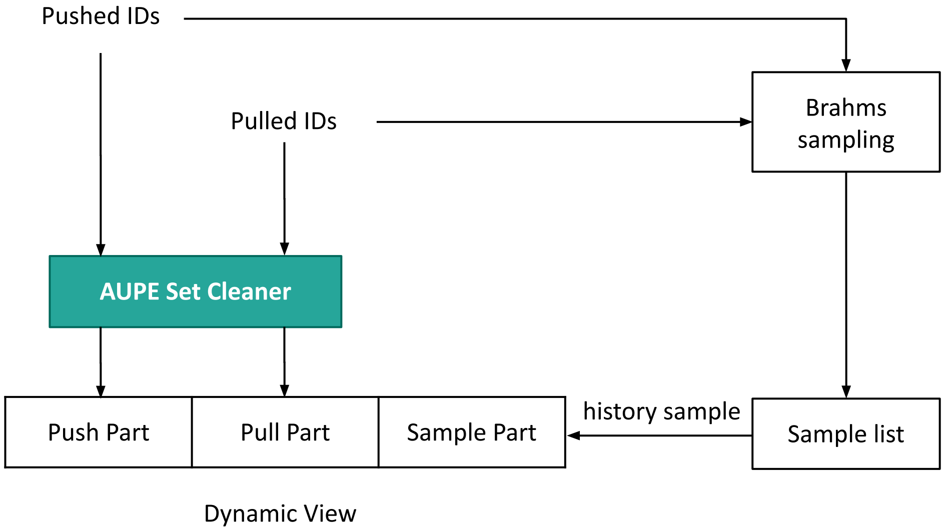
AUPE Protocol

- 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

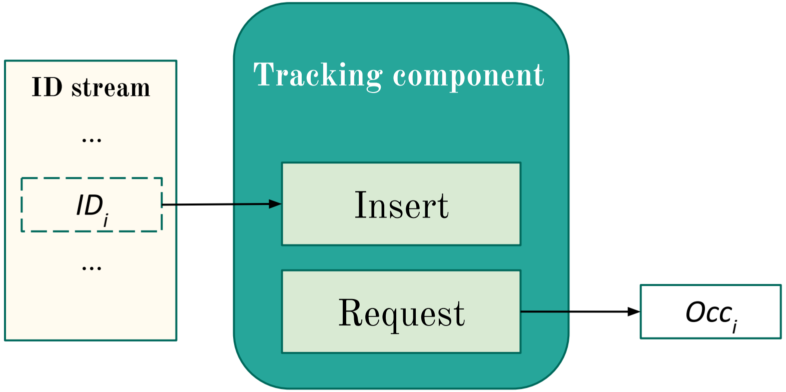
Tracking component

- Record occurrences of received IDs in a tracking data-structure
  - **Key-value store**
  - **Sketch:** Fixed-size data-structure for estimating occurrences

AUPE Set Cleaner

Tracking component

- **Insert** received Ids
- **Request** Id occurrence



Occurrence of node i (real or estimated): Occ_i

AUPE Set Cleaner

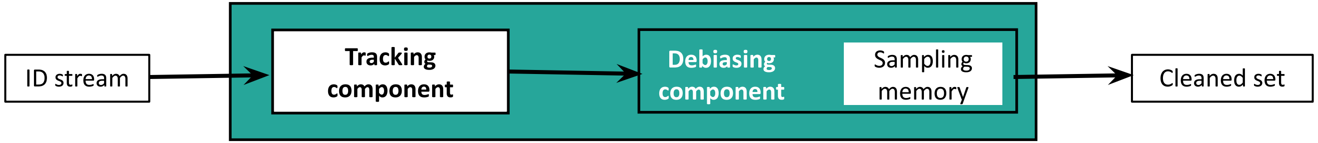
Debiasing component

- Transforms received stream into a more uniform one
- Probability of inserting each item

Probability of insertion of item i : P_i
Minimum of all occurrences : min
Outputed occurrence of node i : Occ_i

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

AUPE Set Cleaner review



Increase of Brahms tolerance by up to 60%

AUPE Secret Collaborative Debiasing 🤝

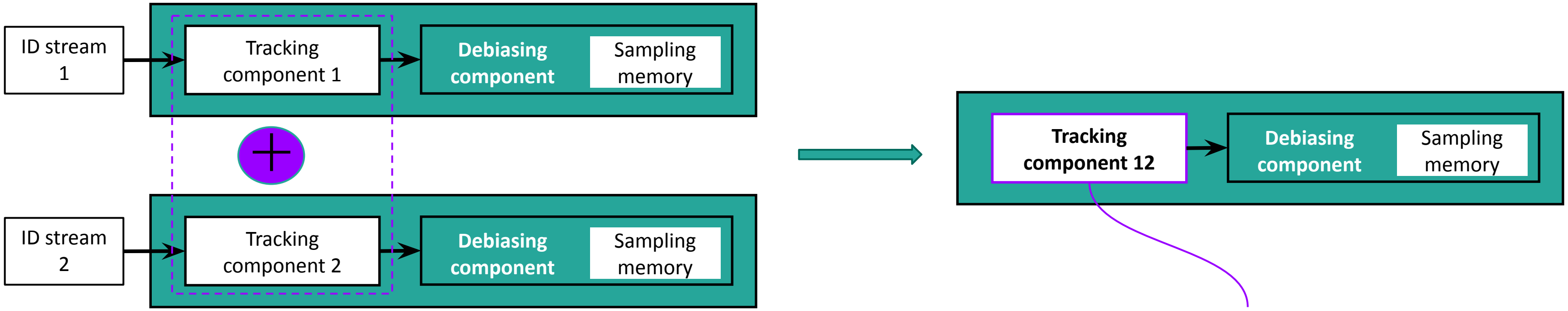
- System is equipped with **Trusted nodes**
  - Based on TEE technology: authenticity of the code
  - **Secure mutual authentication** to recognize trusted peers

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



Combined knowledge of streams 1 and 2

AUPE Secret Collaborative Debiasing 🤝

- **Trusted peer list**
  - 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** (Merge)?
- Compare to **Brahms**, **Basalt**

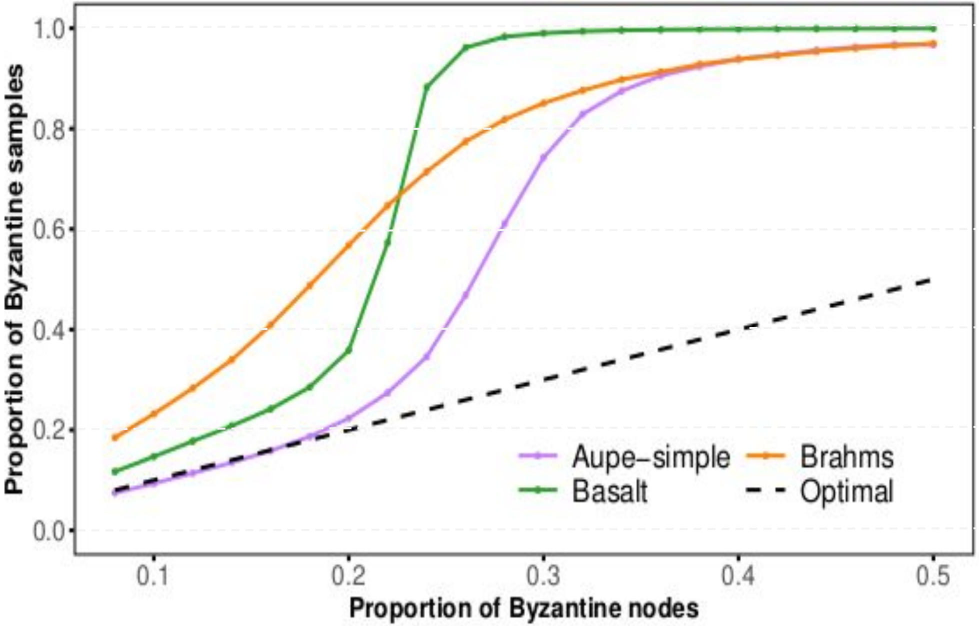
Experimental evaluation

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

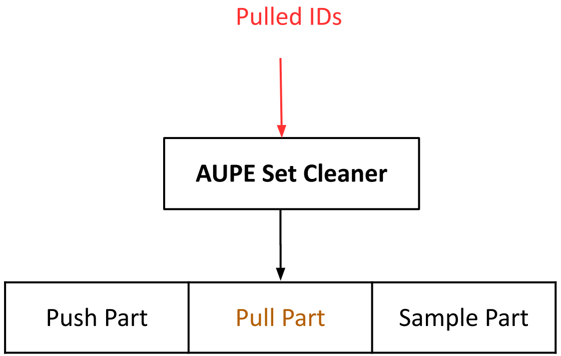
System Tolerance improvement

Aupe-simple

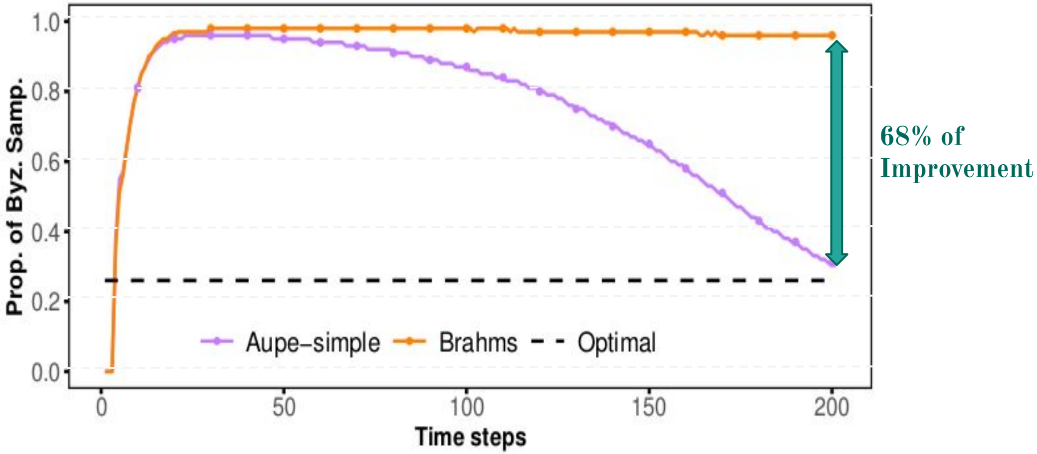


View parts tolerance improvement

Aupe-simple



f=26%

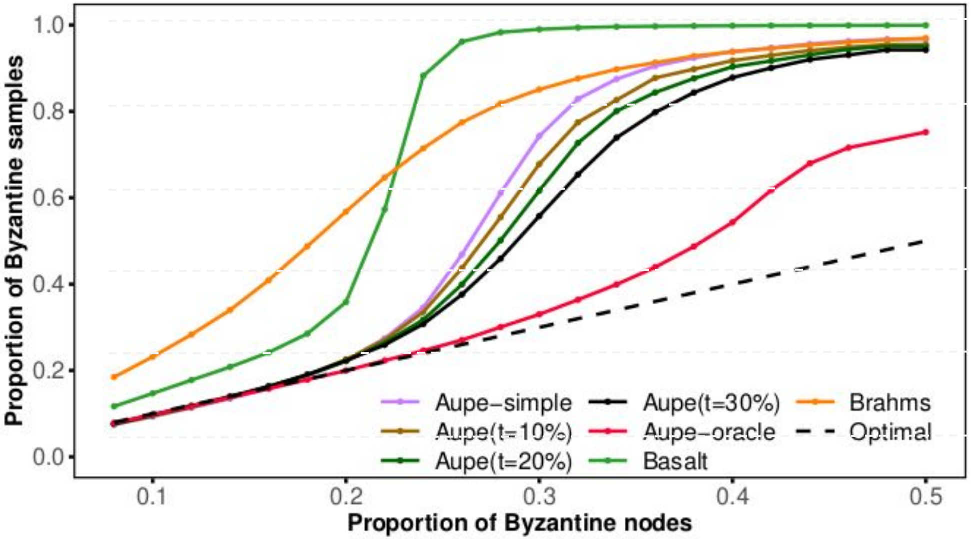


View' Pull part

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

LABORATOIRE
BORDELAIS
DE RECHERCHE
EN INFORMATIQUE
LaBRI

cnrs

université
de BORDEAUX

Looking for research opportunities (academic or industrial) in systems, cloud, and related or interdisciplinary areas.

augusta.mukam@u-bordeaux.fr

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.*