

# Authentication-free P2P fault-tolerance



Wojciech Galuba, Karl Aberer  
EPF Lausanne, Switzerland

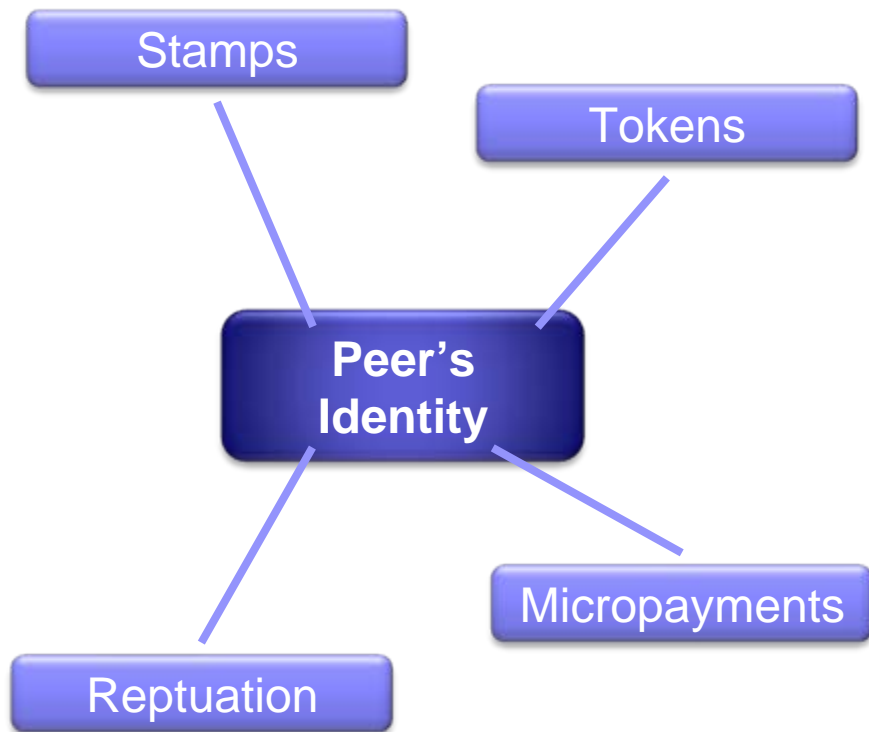


Zoran Despotovic, Wolfgang Kellerer  
DoCoMo Euro-Labs, Munich, Germany

# What is the problem?

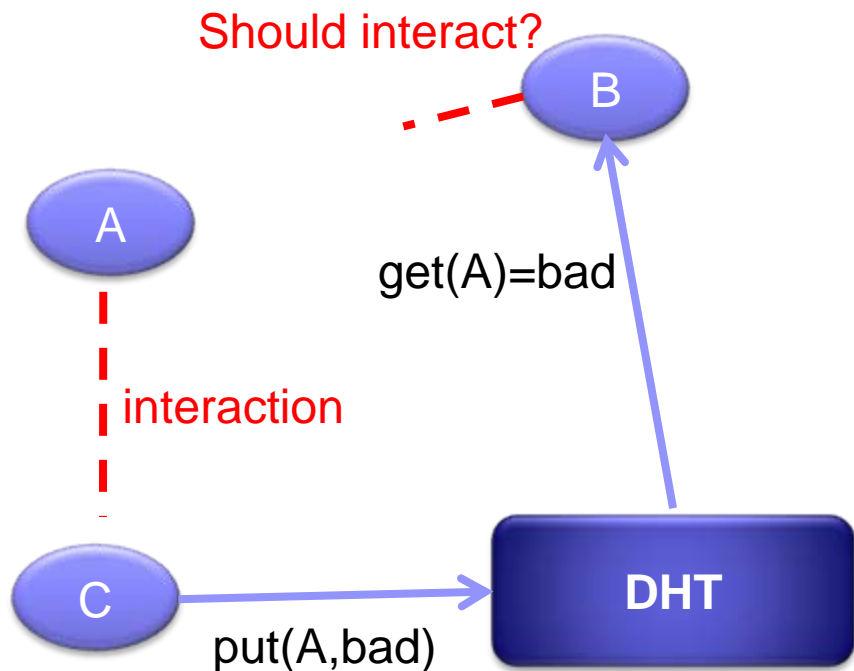
- Typically each peer is owned by a different user
- The user has full control of the peer
  - can arbitrarily modify peer software/hardware
- Users can be:
  - selfish – being clients but not servers
    - P2P cooperation
  - malicious – trying to subvert the network
    - P2P protocol security
- Peers can also fail independently of user actions
  - classical **fault tolerance**
- **We address all these problems in a single framework**

# Existing solutions: problem #1



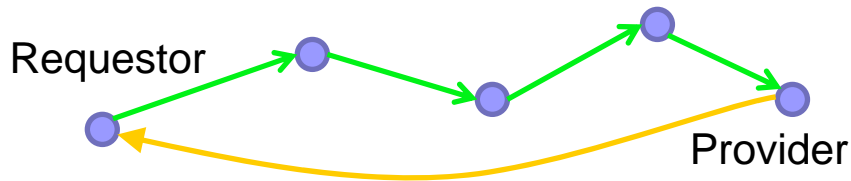
- **Global authentication required**
  - centralized: PKI with CAs
  - decentralized: PGP
- Impractical in large-scale P2P systems
- Privacy ramifications
- We do not rely on global authentication

# Existing solutions: problem #2



- **Shared storage**
  - for e.g. reputation
  - assumed to be reliable
- In practice DHTs used
  - DHTs only resilient to churn
  - Byzantine fault-tolerant routing and storage still a problem
- In our solution: no storage

# Service location & provisioning



## Service location

- locate the service provider in the P2P network
- failures: dropped requests, misrouted requests

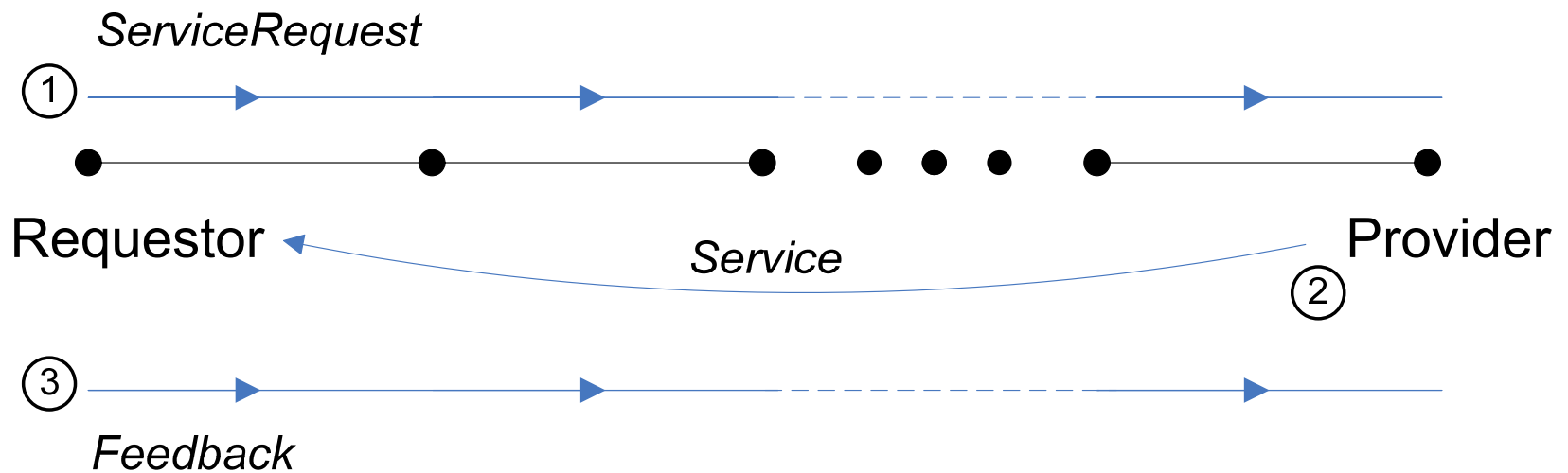
## Service provisioning

- provide the service to the requestor
- failures: service not provided or bad quality

## What are the services?

- File download
  - (key, value) pair lookup/storage
  - Delivery of an overlay message to destination
- 
- Existing approaches: fault tolerance problems in **service location** and **service provisioning** considered separately
  - **We take the end-to-end approach: consider these problems together**

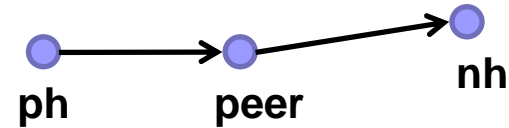
# Forward feedback protocol



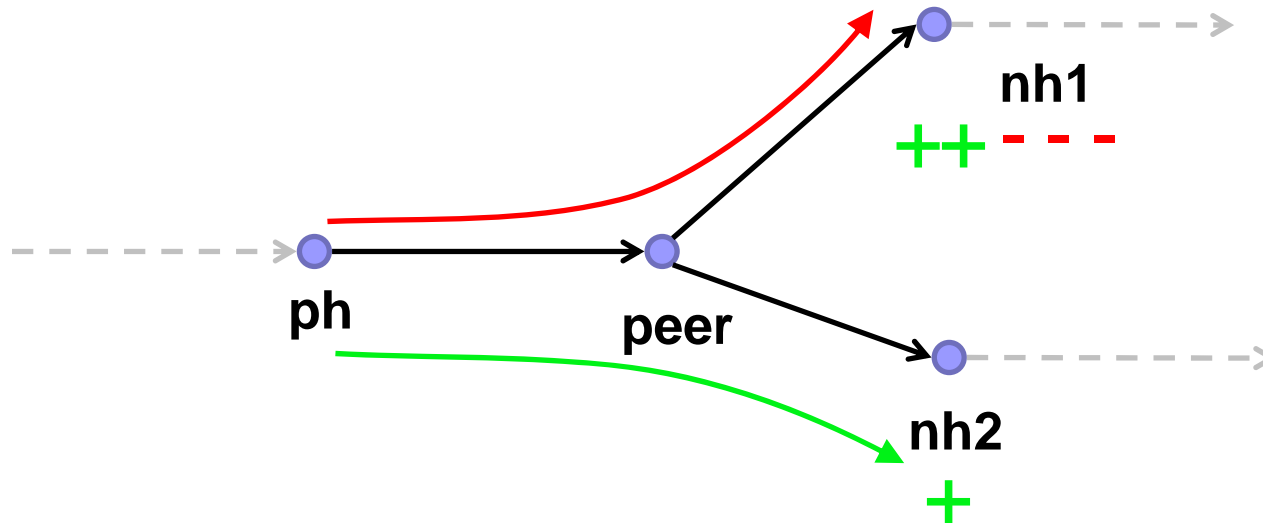
- Requestor estimates the quality of the provided service
  - decision binary: good or bad
- **Feedback follows the same path as the request**
- Feedback is obligatory, no feedback = bad feedback

# A peer on the path

- Knows only its neighbors
  - typically: overlay neighbors
  - no need to authenticate new neighbors
- As feedback arrives, gathers observations (**ph,nh,sd,b**)
  - **ph** – previous hop, **nh** – next hop, **sd** – service descriptor
  - **b** – Boolean true if feedback positive
- For an incoming service request (**ph,sd**)
  - picks **nh** that maximizes the **probability** that service provisioning will be successful
- Peer learns the **probability** based on observations
  - Boolean function learning, the p-concepts algorithm from AI

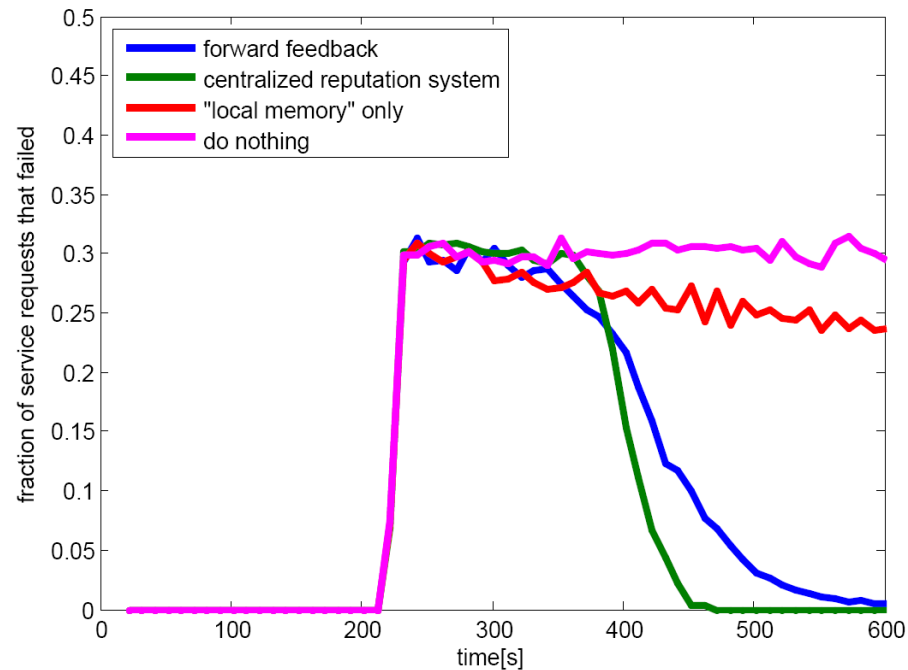


# The FF protocol in action



- **nh1** has history of success but starts failing
- peer switches to **nh2**
- **Collective effect in network: the root cause of a failure receives the most negative feedback**

# Convergence rate



- At 200s mark: 30% of nodes become droppers
- Convergence rate on par with a centralized reputation system

# Lying & feedback mutation

- What if peers
  - lie about the quality of received service?
  - mutate the feedback messages when forwarding?
- How to tolerate?
  - No additional countermeasures necessary besides standard FF
  - False negative feedback eliminated by failure association with the previous hop
  - All false feedback „outvoted” by true feedback
- **In practice tolerance up to 25% liars in the network**
  - Under a coordinated, simultaneous lying attack

# Other FF protocol properties

- Resilient to routing (service location) failures
  - Message dropping
  - Misrouting → TTL exhaustion → timeouts, equiv. to dropping
  - Takes advantage of routing path redundancy
  - Tolerance up to 60% of droppers
- Takes advantage of service replicas
  - Avoids faulty replicas, reroutes to good replicas
- Learns the nature of the failures of the neighbors
  - Which service requests cause problems for which neighbors?
- Resilient to churn
  - Churn causes negative feedback
  - Introduces random noise to learning, well tolerated

# The road ahead

- Introduce a deterrent for selfish behavior
  - Early Tit-For-Tat reciprocation experiments promising
- Test under different workloads
  - Esp. skewed requested service distribution
- Consider Sybil and Eclipse attacks
- First live deployments on PlanetLab
  - Application: secure, delay-minimizing routing in DHTs
  - Measure overheads

# Summary

- The protocol is lightweight
  - Only single bit feedback is exchanged
  - No shared storage
  - No global authentication infrastructure
- Each peer learns locally and independently
- The network learns as a whole to:
  - Prevent bad service provisioning
  - Route around droppers
  - Locate non-faulty replicas
  - Tolerate liars
- Our solution can be applied to any system with recursive routing:
  - Most modern structured overlays
  - DHTs
  - Ad-hoc networks