Gossip-based peer sampling

Mateusz Fedoryszak

on the base of

The fount of all gossip

- Each node has a part of the overall knowledge
- Information periodically exchanged
- High scalability and fault tolerance
Who is my neighbour?

• We need a method of sampling from a set of nodes with an uniform distribution.
• Implement as a gossip-based protocol itself.
• Create a generic protocol, then instantiate and evaluate variations.
The knowledge

• Each node has a list of $c$ descriptors.
• A descriptor is a pair of peer’s IP address and descriptor’s age.
• During information exchange, a node sends its own IP address with age 0 and $c/2 - 1$ of other descriptors it has, randomly selected, not using $H$ oldest.
The cycle

• During each cycle a node initiates just one information exchange.

• May receive many exchange requests.

• At the end of the cycle, all descriptors’ ages are incremented.
Merging

• Add received descriptors to your own view
• Remove duplicates (leave fresher descriptors)
• Remove at most $H$ oldest items
• Remove at most $S$ items sent to a peer

• A care is taken to make view eventually contain exactly $c$ items.
Parameters

• Peer selection
  – selectPeer()

• View selection
  – S – swapping
  – H – healing

• View propagation
  – Push
  – Pull
Is it cool enough?

• Randomness
• Load balancing
• Fault tolerance
Rolling the dice

- Treat sampled peers as a number sequence

- Result: only one failing test
The big picture

• Treat the network as a directed graph
• Vertices ≡ Nodes
• There is an edge \((a, b)\) iff \(a\) stores the descriptor of \(b\)
Convergence

• Initial overlay
  – Growing
  – Lattice
  – Random

• All scenarios lead to a consistent network except push protocol with growing scenario
Do pull!

![Graph showing maximal indegree over cycles]

- push protocols
- pushpull protocols
Indegree standard deviation
Converged indegree distribution
Won’t it blow up?

• Catastrophic failure
• Churn
  – 1%
  – 30%
• Bootstrappling
  – Central
  – Random
Cluster partition

![Graph showing the effect of removing nodes on the largest cluster size.](image_url)
Catastrophic failure recovery
1% churn – degree standard deviation
1% churn – dead links
30% churn
Findings

• It works well
• Use push and pull
• Swapper for good load balancing
• Healer for fault tolerance