Toward Fault-tolerant P2P Systems: Constructing a Stable Virtual Peer from Multiple Unstable Peers

Kota Abe, Tatsuya Ueda (Presenter), Masanori Shikano, Hayato Ishibashi and Toshio Matsuura
Osaka City University

14/Oct/2009 AP2PS2009
Background

- P2P systems pros and cons
  - pros: scalability, no single point of failure, etc.
  - cons: hard to implement!
    - detect remote peer failure
    - replicate data over multiple peers
    - manage multiple pointers to backup peers

- Implementing these measures is delicate work and troublesome burden for developers

GOAL!

Implement a reliable layer for fault tolerant P2P systems
Our Approach

- Virtual Peer (VP)
  - Group multiple unstable peers to form a stable virtual peer (redundant system)
A virtual peer consists of multiple member peers
A P2P application runs on a virtual peer as a virtual process
Failed member peer is replaced with another (non-failed) one
A virtual process is fault-tolerant
- It does not fail even if some part of the member peers fail
- Application developers do not need to take care of peer failure
Issues to solve

1. How to achieve fault-tolerance of a virtual process?
2. How to ensure identical message sequences?
3. How to handle peer failure?
4. How to communicate with a remote virtual peer?
1. Achieving fault-tolerance of virtual process

- The state of a virtual process must be replicated over multiple member peers.
- Each member peer simultaneously and redundantly executes the same application, as a process.
- To maintain the state of each process identical:
  - A process must be a state machine
    - its state must be changed only by external messages
  - Also, each process receives the identical message sequence (aka atomic broadcasting)
- Merit: application programs can be quite simple
  - Just process the received messages in order
2. Ensuring identical message sequences

- To implement atomic broadcast, the **Paxos consensus algorithm** is used

- **Paxos**
  - Distributed algorithm to form a consensus between multiple nodes (peers) on an unreliable network
  - Only a dedicated **leader peer** can propose values
    - The leader is elected by using a leader election algorithm
  - All peers eventually choose an identical value
  - Majority agreement is required

- All the member peers in VP execute Paxos algorithm
  - External messages sent to a VP are processed by the Paxos algorithm to be identically ordered
3. Handling peer failure

- Failed member peer must be replaced to keep the number of the peers constant
  - Otherwise the VP eventually will not be functional because majority agreement is required by Paxos

- All the member peers must have a consistent view of membership configuration

- Paxos is also used to update a member configuration without losing consistency
3. Handling peer failure (Cont'd)

- The leader peer chooses another peer \( p \) from the P2P network
  - If leader peer fails, new leader is elected
- The leader peer proposes a peer configuration change
- \( p \) executes the same process
  - The state must be same
  - Process migration technique is used
- Note that the majority of member peers must be alive during this replacing sequence
How to deliver messages to VPs
  ◦ Member peers are not fixed!

Solution: Use ALM (Application Level Multicast)
  ◦ Each VP has a dedicated ALM group
    • All member peers join in
  ◦ Messages sent to a VP are multicast to the group
  ◦ We have implemented ALM by using range queries on Skip Graph
Our implementation: musasabi

- A platform for implementing P2P services
  - Implemented in Java
- Each peer executes a musasabi instance
- An application program written in Java can be executed on musasabi
- Java sandbox mechanism is used to protect a local node
- musasabi uses PIAX for P2P networking
  - PIAX provides Skip Graph, ALM (over Skip Graph) etc.

Configuration of musasabi

- Base Operating System
  - Windows, MacOS X, Linux ...
- Java VM
  - musasabi
  - PIAX
Process migration in musasabi

- musasabi supports strong mobility
  - Transfer the program, data and execution context (thread stack and program counter)
  - Not easy in Java (not supported by the standard JVMs)
  - Some implementations use customized JVMs or native libraries (not portable)
    - Not suitable for P2P systems!

- Implementation of strong mobility in musasabi
  - Use Apache Javaflow library
    - Javaflow allows to capture and resume the execution context
    - Captured contexts can be transferred to a remote node!
    - Javaflow uses byte code translation technique and thus works on the standard JVMs
A VP fails if a majority of member peers fail.
- Maximum time to replace a failed peer is 60 sec.
- Each peer fails independently.
- Intervals of peer failure are exponentially distributed.
- Peer failure rate: 50% of peers fail in an hour.

Reliability of VP

Elapsed time (days)

7 peers
3 peers
5 peers
7 peers are enough in this case

7 peers are enough in this case.
Relation between MTTF (Mean Time To Failure) of a VP and # of its member peer is analyzed

- Each peer fails independently
- Intervals of peer failure are exponentially distributed
- Maximum time to replace a failed peer is 60sec
- Peer failure rate is varied (x-axis)

Even in excessive peer failure environment, VP is stable if it has enough member peers
Conclusion and Future work

- We proposed a novel method to construct a stable virtual peer from multiple unstable peers
  - Integrate the Paxos consensus algorithm, process migration technique and ALM
  - An application running on a VP virtually does not fail
  - Application programs can be quite simple

- The method can be used for reducing development costs, and for improving stability, of P2P systems

- Future work
  - Improve the method for choosing good member peers
  - Investigate and improve security issues of VPs
  - Evaluate the method on the Internet
Questions?

This research was partially supported by National Institute of Information and Communication Technology (NICT), Japan