Practical Aspects of IP–based Take–Over Mechanisms

Christof Fetzer, Neeraj Suri
AT&T Labs–Research, Florham–Park, NJ
TU Darmstadt, Germany
Motivation

• **Problem:**
  What is a good way to increase the availability of wide-area services by handling server failures?

• **Holy Grail:**
  Transparent replication?
  (Transparent: client service sustained & replicated without any (a) client action or (b) modification of C/S code)
Outline

• Classical Replication: Problems? + Possible Solutions

• Alternatives: Simple Fail–Over
  – Stateless Basis
  – Co–ordination
  – Load Balancing/Shedding
Transparent Replication

client → request → Server → client

Fetzer, Suri
Transparent Replication

client → request → Server → client

Fetzer, Suri
Transparent Replication

client → request → Server → client

Service

Fetzer, Suri
Transparent Replication

Client

Server

Server

Request

Atomic broadcast

Fetzer, Suri
Protocol Issues

client

request via TCP or UDP

Server

Fetzer, Suri
Potential Solution: Atomic Broadcast Tunnel

Translate TCP to atomic broadcast

Fetzer, Suri
Administrative Issue

Fetzer, Suri

client

client

Server

Server
Network Performance

client

low throughput / high latency

high throughput / low latency

Server

Server

Fetzer, Suri
Server Side Broadcast

Client

Request via TCP

Service

Server

Server

client

client

client

Fetzer, Suri
Total Ordering Needed?
Total Ordering Not Sufficient
Leader/Follower Protocol

Fetzer, Suri
Transparent 1–n TCP

Replication of input queues sufficient

Fetzer, Suri
Target Application Domain

Client

... Client

Service

SAMBA Server

SAMBA Server

Fetzer, Suri
Issues

• Coping with non-determinism is difficult. Leader/Follower protocol is
  – not easy to implement
  – not easy to keep up-to-date
  – not easy to harden

• Software bugs: all replicas fail at same point

• Clients have failure detectors – how do they fit in?

• Different apps detect failures at different times?

• Support for service/server specific clients?
• Is transparent replication all that great even if we can implement it?

• As clients can detect failures and reconnect is simpler IP take-over more practical?
Approach: Simple Fail–Over

- Internet unreliable:
  - loss of connection to server must be expected
  - many clients try to reconnect

- For many services, a simple fail-over might be sufficient:
  - Single host system:
    - typically, a crashed server is automatically restarted,
  - Extension of multi-host system:
    - If host crashes, server is restarted on a different host
Stateless Servers

client

Server

inter-request state

dependable storage system

request 1 3 reply

Fetzer, Suri
Recovery

Fetzer, Suri

Dependable storage system

Inter-request state

Request

Server

Client
Simple IP Take–Over

Request via TCP or UDP
Simple IP Take–Over

client

IP Addr
Server

IP Addr
Server

Fetzer, Suri
Implementation

Only leader has the right to use the service IP address

Fetzer, Suri
Requirements

- At most one leader at a time
- Infinitely often there is a leader
- Correct server with the highest priority becomes leader (unless a failure occurs)
- Only a failure can lead to a leader change [leader stability]

Fetzer, Suri
Load Balancing & Fail–Over

• For small work loads, a single server solution is typically sufficient
• For large work loads, load balancing combined with fail–over might be needed.
Load Balancing

client \rightarrow \text{Load balancer} \rightarrow \text{Server} \rightarrow \text{client}

\ldots

Fetzer, Suri
Issues

• Dynamic mapping of clients to servers:
  – Load balancer needs to keep state (fail-over more difficult)

$ Can be expensive (may need to pay for dynamically changing load balancer mapping)

• Static mapping of clients to server:
  – No dynamic adjustment of load possible
IP Take-Over Solution

Static mapping of clients to virtual IP addresses

Fetzer, Suri
Dynamic Mapping

- Each virtual IP address mapped on at most one server
- Load shedding by not assigning all virtual IP addresses
Membership Based Solution

- Server is assigned virtual IP addresses based on current membership
- Possible Extension:
  - Add load info to membership view
  - Load info facilitates dynamic load balancing
Example

Servers | T1 | T2
--- | --- | ---
S1 | IP1 | IP1, IP3 | IP1, IP3
S2 | IP2 | IP2 | Load shedding
S3 | IP3 | Assigned IP address(es) | Assigned IP address(es)

V={S1, S2, S3} | V={S1, S2} | V={S1}
Requirements

• Synchronized clocks
• Agreement on membership view and time when new view becomes valid
• Each host assumes IP addresses at change of membership:
  \[ f(\text{view}, \text{time}, \text{IP}) \Rightarrow \text{host} \]
Conclusion

• IP based replication mechanisms have advantages for classical client transparency
• Transparent replication only makes sense for certain services
• Stateless server designs are simple and provide good fail–over properties
• Future work:
  – Add load information hooks to membership protocol