Failure Detectors and Membership Protocols

March 18, 2010

Acknowledgments: Dongyun Jin and Thuy Nguyen

Target Settings

- Process ‘group’-based systems
  - Clouds/Datacenters
  - Replicated servers
  - Distributed databases
- Crash-stop/Fail-stop process failures

Group Membership Service

Application Queries
- e.g., gossip, DHT

Group Membership List
-joins, leaves, failures

Unreliable Communication

Two sub-protocols

Application Process $p_i$

Dissemination

Failure Detector

Almost-Complete list (focus of this talk)
- Virtual synchrony, Gossip-style, SWIM, ...
- Or Partial-random list (other papers)
  - SCAMP, T-MAN, Cyclon,...

Large Group: Scalability A

Goal

Process Group
- "Members"

$1000’s$ of processes

Unreliable Communication Network

Group Membership Protocol

Application Process $p_i$

$pi$ crashed

Failure Detector

Some process finds out quickly

Unreliable Communication Network

Crash-stop Failures only
I. pj crashes
- Nothing we can do about it!
- A frequent occurrence
- Common case rather than exception

II. Distributed Failure Detectors: Properties
- Completeness = each failure is detected
- Accuracy = there is no mistaken detection
- Speed – Time to first detection of a failure
- Scale – Equal Load on each member
  – Network Message Load

Distributed Failure Detectors: Properties
- Completeness
- Accuracy
- Speed – Time to first detection of a failure
- Scale – Equal Load on each member
  – Network Message Load

What Real Failure Detectors Prefer
- Completeness
  Guaranteed
- Accuracy
  Partial/Probabilistic guarantee
- Speed – Time to first detection of a failure
- Scale – Equal Load on each member
  – Network Message Load

Failure Detector Properties
- Completeness
  Guaranteed
- Accuracy
  Partial/Probabilistic guarantee
- Speed – Time to first detection of a failure
- Scale – Equal Load on each member
  – Network Message Load

Failure Detector Properties
- Completeness
  Guaranteed
- Accuracy
  Partial/Probabilistic guarantee
- Speed – Time to first detection of a failure
- Scale – Equal Load on each member
  – Network Message Load
  – No bottlenecks/single failure point
### Failure Detector Properties

- **Completeness**
- **Accuracy**
- **Speed**
  - Time to first detection of a failure
- **Scale**
  - Equal Load on each member
  - Network Message Load

In spite of arbitrary simultaneous process failures

### Centralized Heartbeating

- Heartbeats sent periodically
- If heartbeat not received from $pi$ within timeout, mark $pi$ as failed

### Ring Heartbeating

- Unpredictable on simultaneous multiple failures

### All-to-All Heartbeating

- Equal load per member

### Gossip-style Heartbeating

- Good accuracy properties

### Gossip-Style Failure Detection

- Protocol:
  - Nodes periodically gossip their membership list
  - On receipt, the local membership list is updated

Current time: 70 at node 2
(asynchronous clocks)
Gossip-Style Failure Detection

- If the heartbeat has not increased for more than $T_{\text{fail}}$ seconds, the member is considered failed.
- And after $T_{\text{cleanup}}$ seconds, it will delete the member from the list.
- Why two different timeouts?

Multi-level Gossiping

- Network topology is hierarchical.
- Random gossip target selection => core routers face $O(N)$ load. (Why?)
  - Fix: Select gossip target in subnet $I$, which contains $n_i$ nodes, with probability $1/n_i$.
  - Router load = $O(1)$.
  - Dissemination time = $O(\log(N))$.
  - Why?
  - What about latency for multi-level topologies? (Gupta et al, TPDS 06)

Analysis/Discussion

- What happens if gossip period $T_{\text{gossip}}$ is decreased?
- A single heartbeat takes $O(\log(N))$ time to propagate. So: $N$ heartbeats take:
  - $O(\log(N))$ time to propagate, if bandwidth allowed per node are allowed to be $O(N)$
  - $O(N \log(N))$ time to propagate, if bandwidth allowed per node is only $O(1)$
  - What about $O(k)$ bandwidth?
- What happens to $P_{\text{mistake}}$ (false positive rate) as $T_{\text{fail}}, T_{\text{cleanup}}$ is increased?
- Tradeoff: False positive rate vs. detection time.

Simulations

- As # members increases, the detection time increases.
- As requirement is loosened, the detection time decreases.
- As # failed members increases, the detection time increases significantly.
- The algorithm is resilient to message loss.

Failure Detector Properties …

- Completeness
- Accuracy
- Speed
  - Time to first detection of a failure
- Scale
  - Equal Load on each member
  - Network Message Load
...Are application-defined Requirements

- Completeness
- Accuracy
- Speed
  - Time to first detection of a failure
- Scale
  - Equal Load on each member
  - Network Message Load

Guarantee always
Probability $PM(T)$
$T$ time units

Scale
- Equal Load on each member
- Network Message Load

25

All-to-All Heartbeating

Every $T$ units
$L = N/T$

$pi$, Heartbeat Seq. $l++$

27

Gossip-style Heartbeating

$L = N/tg = N \log N / T$

Every $tg$ units
$T = \log N \cdot tg$

Array of Heartbeat Seq. $l$
for member subset

28

What's the Best/Optimal we can do?

- Worst case load $L^*$
  - as a function of $T, PM(T), N$
  - Independent Message Loss probability $p_{ml}$

$L^* = \frac{\log PM(T)}{\log(p_{ml})} / T$
(proof in PODC 01 paper)

29

Heartbeating

- Optimal $L$ is independent of $N$
- All-to-all and gossip-based: sub-optimal
  - $L = O(N/T)$
  - try to achieve simultaneous detection at all processes
  - fail to distinguish Failure Detection and Dissemination components

Key:
- Separate the two components
- Use a non heartbeat-based Failure Detection Component

30
### SWIM Failure Detector Protocol

- **Protocol period**: $T'$ time units
- **Random processes**: $K$ processes
- **Random ping**: $p_i$
- **Random ping-req**: $X_i$
- **Random ack**: $p_j$

### SWIM versus Heartbeating

#### First Detection Time
- **SWIM**: Expected $\frac{e}{e-1}$ periods
- **Heartbeating**: Constant (independent of group size)

#### Process Load
- **SWIM**: Constant per period
- **Heartbeating**: $< 8 L^*$ for 15% loss

#### False Positive Rate
- **SWIM**: Tunable
- **Heartbeating**: Falls exponentially as load is scaled

#### Completeness
- **SWIM**: Deterministic time-bounded
- **Heartbeating**: Within $O(\log(N))$ periods w.h.p.

### Detection Time

- **Prob. of being pinged in $T'$**: $1 - \left(1 - \frac{1}{N}\right)^{K-1} = 1 - e^{-3}$
- **$E[T]$**: $T' \cdot \frac{e}{e-1}$
- **Completeness**: Any alive member detects failure
  - Within worst case $O(\log(N))$ protocol periods

### Accuracy, Load

- **$PM(T)$ is exponential in $K$. Also depends on $pml$ (and $pf$)**
  - See paper
- **$L / L^* < 28$**
- **$L / L^* < 8$** for up to 15% loss rates

### III. Dissemination

- **Some process finds out quickly**
- **Unreliable Communication Network**
- **HOW?**
Dissemination Options

- Multicast (Hardware / IP)
  - unreliable
  - multiple simultaneous multicasts
- Point-to-point (TCP / UDP)
  - expensive
- Zero extra messages: Piggyback on Failure Detector messages
  - Infection-style Dissemination

Infection-style Dissemination

- Epidemic style dissemination
  - After $\log(N)$ protocol periods, $\sqrt{N}$ processes would not have heard about an update
- Maintain a buffer of recently joined/evicted processes
  - Piggyback from this buffer
  - Prefer recent updates
- Buffer elements are garbage collected after a while
  - After $\log(N)$ protocol periods; this defines weak consistency

Suspicion Mechanism

- False detections, due to
  - Perturbed processes
  - Packet losses, e.g., from congestion
- Indirect pinging may not solve the problem
  - e.g., correlated message losses near pinged host
- Key: suspect a process before declaring it as failed in the group

Suspicion Mechanism

- Distinguish multiple suspicions of a process
  - Per-process incarnation number
  - inc # for $pi$ can be incremented only by $pi$
    - e.g., when it receives a (Suspect, $pi$) message
  - Somewhat similar to DSDV
- Precedence rules for (Alive, inc #), (Suspect inc #), (Failed, inc #)
  - See paper
**Time-bounded Completeness**

- Key: select each membership element once as a ping target in a traversal
  - Round-robin pinging
  - Random permutation of list after each traversal
- Each failure is detected in worst case 2N-1 (local) protocol periods
- Preserves FD properties

**Results from an Implementation**

- Current implementation
  - Win2K, uses Winsock 2
  - Uses only UDP messaging
  - 900 semicolons of code (including testing)
- Experimental platform
  - Galaxy cluster: diverse collection of commodity PCs
  - 100 Mbps Ethernet
- Default protocol settings
  - Protocol period=2 s; K=1; G.C. and Suspicion timeouts=3ceil[log(N+1)]
- No partial membership lists observed in experiments

---

**Per-process Send and Receive Loads**

- Independent of group size

**Time to First Detection of a process failure**

- Apparently uncorrelated to group size

**Membership Update Dissemination Time**

- Low at high group sizes
More discussion points

- It turns out that with a partial list that is uniformly random, gossiping retains same properties as with complete lists
  - Why?
  - Partial membership protocols
    - SCAMP, Cyclon, TMAN, ...
- Gossip-style failure detection underlies
  - Astrolabe
  - Amazon EC2/S3 (rumored!)
- SWIM used in
  - CoralCDN/Oasis anycast service: http://oasis.coralcdn.org
  - Mike Freedman used suspicion mechanism to blacklist frequently-failing nodes

Questions