On Failure Detection Algorithms in Overlay Networks

INFOCOM‘ 2005

2005/09/30
Outline

- Introduction
- Network Model
- Keep-Alive Algorithms
- Evaluation
- Conclusion
Overlay networks are seen as an excellent platform for large scale distributed systems -- one reason is resilience: in three aspects

- Data replication
- Routing recovery
- Static resilience

Failure detection algorithms can be classified as

- Active approach: periodically send keep-alive messages
- Passive approach:
  - Use data packets to convey aliveness information,
  - Inadequate in several situations
  - Can be viewed as an optimization of active approach when data traffic is present

rely on accurate and timely detection of node failures
Two classes of active keep-alive approaches

- **Baseline**: each node independently decides whether its neighbors are alive or not
- **Sharing**: nodes share aliveness information and thus can reduce the failure detection time

There is a tradeoff between

- **Minimizing the failure detection time** may increase the **Probability of false positive** (making a false detection)
- The lower the **failure detection time**, the higher the cost of **control overhead**
- **Packet loss rate**: packets are lost due to forwarding to a failed neighbor
Goals of this paper

- To examine how keep-alive algorithms can detect failures as soon as possible when a node can no longer communicate with a neighbor.
- To examine how the design of various keep-alive approaches affect their performance in:
  - Detection time
  - Probability of false positive
  - Control overhead
  - Packet loss rate
Network Model

- An overlay network with $n$ nodes
  - $N(A)$: Neighbor set of $A$ -- Each node $A$ knows $d$ other nodes in the network
  - Node $A$ maintains its neighbor set by sending ack "are you alive" probes every $\Delta$ seconds to each neighbor
  - Node failure: assume nodes fail in a failstop manner; assume nodes join according to a Poisson process and fail according to an exponential distribution
  - Packet loss: assume due to
    - Transient problem such as network congestion
    - Network link failures
  - Propagation delay: a node consider a probe lost if it does not receive an ack within $T_{to}$ seconds
  - Probe traffic: a node must bound the aggregate rate of probes received to some reasonable rate $R$
Keep-A live Algorithms

- Five keep-alive algorithms are presented
  - Not to model a specific keep-alive algorithm but rather to capture the essential aspects towards failure detection

<table>
<thead>
<tr>
<th>Axes</th>
<th>Baseline</th>
<th>SN+ BPTR</th>
<th>SN</th>
<th>SNP+ BPTR</th>
<th>SNP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gossip vs. probe</td>
<td>Probe</td>
<td>Probe</td>
<td>Probe</td>
<td>Probe</td>
<td>Probe</td>
</tr>
<tr>
<td>Node vs. net failures</td>
<td>both</td>
<td>both</td>
<td>both</td>
<td>both</td>
<td>both</td>
</tr>
<tr>
<td>Sharing information</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Neg vs. pos info</td>
<td>-</td>
<td>neg</td>
<td>neg</td>
<td>both</td>
<td>both</td>
</tr>
<tr>
<td>Keep-alive state</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
Baseline

- Node $A$ sends a probe to its neighbor $F$ every $\Delta$ seconds and waits for an ack.
- If a probe is not acknowledged within $T_{to}$ seconds, it is considered lost:
  - Then next probe is sent $T_{qp} (> T_{to})$ seconds after the previous probe, up to a maximum of $c-1$ quick probes.
  - A node removes a neighbor from its routing table after $c$ consecutive timeouts.
Sharing Negative Information with Backpointer State (SN+BPTR)

- Sharing Negative Information with Backpointer State: share negative (node is down) information among nodes who are interested in a particular neighbor
  - **B(F)**: Backpointer of F, the set of nodes which have a node F in their neighbors set
  - When a node in B(F) experiences c consecutive timeouts to F, it sends negative information (boost) to all other nodes in B(F)
  - As in-degree b of a node increases, Δ has to increase proportionally to maintain the aggregate probe rate R
    - Reduce the probability of false positive: impose a constraint such that the time span of the last k boosts must be less than a time window, T_{boost}
Sharing Negative Information (SN)

- When a node $A$ experiences $c$ consecutive timeouts to a neighbor $F$, it sends a boost to its other neighbors.
- A node removes a neighbor from its routing table if it experiences $c$ consecutive timeouts, or receives $k$ consecutive boosts.
Sharing Negative and Positive Information with Backpointer State (SNP+BPTR)

- SNP+BPTR is similar to SN+BPTR, with the addition of sharing positive (node is up) information to reduce the probability of false positive.

- When $A$ receives an ack from $F$ and its boost counter for $F$ is nonzero, it sends this positive information (posinfo) to other backpointers ($B$ and $C$).

- When $B$ receives the posinfo, it resets the boost counter for $F$ to 0.

- When $F$ is up but the path between it and a node is down, the node will still remove $F$ from its routing table because posinfo only resets boost counter and not the timeout counter.
Sharing Negative and Positive Information (SNP)

- SNP is similar to SN, with the addition of positive information to reduce the probability of false positive.
- When A receives an ack from F and its boost counter for F is nonzero, it sends this posinfo to its other neighbor (B and C).
- When B receives the posinfo and has F as a neighbor, it resets the boost counter for F to zero.
Evaluation

- Simulation and experimental results in the context of Chord

Methodology

- Start a Chord network with 2000 nodes
- Key lookups (packets) are initiated from random sources to random keys, timed by a Poisson process at a rate of 200 per second
- Two kinds of loss models are evaluated
  - LM1: packet losses are due to transient network problem, each packet traversing an overlay link is dropped independently with fixed probability $p = 0.4\%$
  - LM2: injected network link failures so that the average unavailability of the path is 1.25\%
LM1 Results

Detection Time Histogram, 2000 node network

median node life time = 30 min

size of neighbor set = 44
Metrics vs. Size of Neighbor Set

Detection Time vs. Size of Neighbor Set, 2000 node network

Bandwidth vs. Size of Neighbor Set, 2000 node network

4.5 times lower than baseline

1.8 times higher than baseline
Metrics vs. Churn Rate

- 3-4 times lower than baseline
- Higher churn rate
- More node failure
- More extraneous boosts
**LM2 Results**

- Simulation with $n=1000$ nodes, mean lifetime = 22 min, $d=128$, and $p=0.05$
- Result for detection time is similar to that under LM1 loss model
Conclusion (1/2)

- **Main findings**
  - **Detection time vs. sharing**
    - No network failure: sharing achieves both lower detection time and control overhead than baseline, with comparable probability of false positive
    - With network failure: sharing improves detection time at the cost of increased control overhead
  - **Detection time vs. size of neighbor set**
    - Improvement in detection time between baseline and sharing becomes more pronounced as size of neighbor set increases
Packet loss vs. size of neighbor set
- Baseline: packet loss rate is a function of detection time, which increases linearly as degree increases if probe bandwidth stays constant
- Sharing: packet loss rate is a function of path length, which decreases as the degree increases

Packet loss rate vs. churn rate
- For a target packet loss rate, sharing of information allows a network to operate at a higher churn rate than baseline