An Autonomic Failure-Detection Algorithm

(1) Bounds worst-case and best-case failure-detection latencies
(2) Bounds resource (bandwidth and processing) consumption devoted to failure-detection
(3) Adjusts worst-case failure-detection latency as system size varies

Kevin Mills, Scott Rose, Stephen Quirolgico, Mackenzie Britton, and Ceryen Tan

January 2004

Scalable Software for Hostile & Volatile Environments
Resource vs. Latency Tradeoff in Two-Way Failure-Detection Systems

- $T_R$ = time of rising heartbeat
- $H_P$ = heartbeat period
- $N$ = number of Monitorables
- $T_F$ = time of falling heartbeat
- $S_R$ = size of rising heartbeat
- $S_F$ = size of falling heartbeat
- $T_R$ = time of rising heartbeat
- $T_T$ = time of falling heartbeat
- $TR + HP$ = rising heartbeat
- $TF + HP$ = falling heartbeat
- $SR$ = size of rising heartbeat
- $SF$ = size of falling heartbeat

Resource Consumption

Bandwidth Usage ($B$)

$B = N (S_R + S_F) / H_P$

Failure-Detection Latency

- $L_{AVG} = H_P / 2$
- $L_{MAX} = H_P$
The Autonomic Algorithm

Define Three Policy Goals:
(1) Worst case avg. failure-detection latency ($L_{WORST}$)
(2) Best case avg. failure-detection latency ($L_{BEST}$)
(3) Allocated Bandwidth ($B_A$)

Initialize Algorithm Parameters:
$H_{MAX} = 2 L_{WORST}$ max. $H_P$
$H_{MIN} = 2 L_{BEST}$ min. $H_P$
$C = B_A/(S_R + S_F)$ max. rate
$N = 0$ # of monitorables

Algorithm Properties:
Varies and bounds $H_P$ and $N$
($H_{MIN} \leq H_P \leq H_{MAX}$)
($0 \leq N \leq N_{MAX}$)
Bounds resource consumption:
~ $C$ heartbeats/sec
~ $B_A$ bytes/sec

MONITORABLE ACTIONS
send Rising Heartbeat to Monitor
do forever
delay $H_P$ seconds
send Rising Heartbeat to Monitor
end do

MONITOR ACTIONS
On Each Rising Heartbeat
if new monitorable then $N++$;
if $H_P > H_{MAX}$
then $N--$;
raise capacity exception;
elseif $H_P < H_{MIN}$
then $H_P = H_{MIN}$;
endif
return $H_P$ to monitorable
Analysis of Algorithm Behavior

Number of Monitorables ($N$)

Heartbeat Period ($H_P$) in Seconds

Bandwidth Consumption ($B$) in B/s

Failure-Detection Latency ($L$) in Seconds
Value of the Algorithm

- Given three policy parameters, the algorithm enforces specified bounds on resource consumption and failure-detection latency, while automatically adjusting worst-case failure-detection latency as system size varies.

- The algorithm is simple to implement and effective in operation.

- The algorithm can be applied in a wide range of distributed object systems to bound inconsistency of cached information about object status.

- The algorithm is especially well-suited to applications where remote objects contact directories and caches periodically to update soft-state information.

- **POSTER:** Algorithm applied in three state-of-the-art service-discovery systems: Jini Network Technology™, the Service Location Protocol (SLP), and Universal Plug-and-Play (UPnP).