A Self-organizing Isolated Anomaly Detection Architecture for Large Scale Systems

E. Anceaume\textsuperscript{1}  E. Le Merrer\textsuperscript{2}  R. Ludinard\textsuperscript{3}  B. Sericola\textsuperscript{3}  G. Straub\textsuperscript{2}

NEM summit 2013
\textsuperscript{1}IRISA/CNRS, \textsuperscript{2}Technicolor, \textsuperscript{3}Inria
1. Problem definition: Isolated error detection
2. A distributed solution
3. A central solution (a word on)
Monitoring

Monitoring a system is the ability of continuously collecting and analyzing relevant information provided by the monitored entities so as to be continuously aware of the system state.

- Focus on **ISP or OTT providers**.
- Clients may experience quality degradation.
- Targets large-scale deployments.
Considered system
Considered system
Quality degradation due to a network failure
Quality degradation due to a service failure
Quality degradation due to an end device failure
Distinguishable?

End device
Network
Service

ISP

Erwan Le Merrer
Isolated Anomaly Detection - NEM summit
Indistinguishable!
Isolated Anomaly Detection Problem

From one end, how to make a distinction between isolated failures and global failures?
Monitoring at the edge
Monitoring at the edge

- Pushing monitoring task at the edge of the network.
- (vs. collecting all logs + process them offline in a datacenter)
- End softwares monitor service qualities.
Monitoring task infinite loop

- **Q(p,t)**
  - Move within the overlay
  - **A(p,t)**
    - Am I alone?
      - yes → **Local failure**
      - no → Network Related failure
      - no → Rise Alert

- if no return to Move within the overlay.
Model

- Time is discretized (e.g. 1mn time slot).
- Nodes self-organize according to their QoE.
- QoE space is tessellated into elementary quality buckets.
- Nodes move in the overlay according to their QoE variation.

![Graph showing QoE of service 1 and service 2]
Seed

A Seed is a bucket (noted $b$) where there are at least $S_{\text{min}}$ nodes.
Definitions

**Cell**
A cell is defined as an hyper-rectangle of buckets, among which at most one is a seed.

Overlay = CAN-like Distributed Hash Table.
On QoE change at node $n$

$n$ moves in the overlay and joins a new seed, containing nodes with close QoE.
Tracking isolated anomalies

\[ \mathcal{H}(b_1, b_2, t - 1) \]

\( n \) increments a value located at \( \mathcal{H}(b_1, b_2, t - 1) \)
An alert is reported by the node hosting the counter if $value \leq \tau$. 
Cost of operation (analytical)

- $N$ nodes
- $D$ services
- $n_i = 1/\rho_i$
- Node join: $\Theta(\log N)$
- Cell split: $\mathcal{O}(D)$
- Node leave: $\Theta(\log N)$
- Cell merge: $2Dn^{D-1}\log_2(n/2)$ messages, with $n = \max_{1 \leq i \leq D} n_i$
- Lookup: $\mathcal{O}(\log N)$

Protocols in OPODIS’12 paper.
Solution 2 (central)

Monitoring at the datacenter
Moni
toring centrally: stream like approach

- Gateways simply push their QoE values frequently.
- Collect point implements the detection algorithm.
- Stream processing like approach.
R-tree based approach

Each received value is placed into a spacial access structure:

In a 2-D space (here we monitor a single service), a point is past & present QoE of a gateway in observed time slot.
Then analyze each leaf:

If grouped points under $R$-balls (quality bucket radius) are less than $\tau$, then isolated errors are detected and notified.
Formal definition of the problem of isolated anomaly detection.

Two solutions, one stream-based (central) and one distributed, that are:

- Dynamic
- Scalable
- Generic (any service representable with QoE/QoS)

Future work:

- Currently implementing/testing the stream-based solution.
Questions ?

PhD work by Romaric Ludinard.


erwan.lemerrer@technicolor.com