Knowledge is Imperfect

ACTING ON STALE, INCONSISTENT OR MISSING DATA

ULF WIGER, FEUERLABS, INC.

© FEUERLABS 2013

Wednesday, 2 October 13
Outline

War stories
No code, no algorithms, no Hadoop

Thoughts
The code may well be broken before it’s even written
Experience
Alaskan Adventure


The core of Command & Control is control of information
“Where are my assets, and what is their status?” (Col Shepherd)

Near Real-time

World-wide

No single point of failure

Pull information from any source
Ericsson adventure

13 years building telephony systems at Ericsson

World’s first carrier-grade voice-over-packet systems
Feuerlabs Adventure—ongoing

“Connecting the Internet of Things™”

Building modern Connected-Device Management services
Traits
C2: Distinctive Challenges

Assume enemy...
actively tries to destroy your infrastructure
actively feeds you misleading information

Deploy anywhere, anytime

Fallback: fully manual

Mess up—people die!
Solutions (then)

No single point of failure
  Full asynchronous replication (40 sites)

Synchronization
  Control access; strict ownership
  Rely on model for manual operation

Split brain
  Site-specific data cached at remote sites

Limited connection speed (down to 19.2 KBps)
  Priority-based replication
Telecom: Special Challenges

Ubiquitous service
People expect it to always work

Emergency calls
Should be serviced even during extreme overload

“User-friendly” failure modes
Few seconds setup time
Echo cancellation, speech quality, tolerable delays

Legacy
Generations of hardware, software, protocols
Device Management Challenges

Information access & quality
- RPC validation
- Config data consistency
- SW status (OTA upgrades)

User requirements unclear

Connection quality/cost
- Remote probes
- Sandboxing/security
- Fail/retry/timeout

Wednesday, 2 October 13
Decision Support Basics

The Four Ws:

Who reported?
What happened?
When did it happen?
Where did it happen?
(The Why is saved for post-mortem)
The Who

Affects our level of trust
Sometimes, deliberate misinformation
Other times, you take what you can get
The What

Surprisingly hard to report sufficient information

- Missing data
- Conflicting data
- Incorrect data

- It could be a message that--
- It could be a fake Russian transmission.

© FEUERLABS 2013
Abstractions

Different views for different roles

Aggregation / Drill-down
Ulf’s Law of Information Management

The key information flow in any organization is bottom-up
Not managers telling workers what they should know

Keep low-level information, aggregate up
Allow digging into details as needed

Many bad decisions are based on missing or misleading data
The ability to shape data for reporting is a power factor
Automation can mitigate this
The ‘What’ for Developers

What are we going to build?
Often surprisingly vague

An organization loses its intuition when the person who has the answer isn’t talking to the person who has the question (Tim Berners Lee, “Weaving the Web”– from memory)
Dealing with requirements

Agile methods great for bottom-up development

Software development is a top-down / bottom-up activity

Tony Hoare’s Turing Award Speech:
   One man/group whose purpose is to understand what is being done, and why
Specifications

STR ::= < Diameter Header: 275, REQ, PXY >

< Session-Id >
{ Origin-Host } 
{ Origin-Realm } 
{ Destination-Realm } 
{ Auth-Application-Id } 
{ Termination-Cause } 
[ User-Name ] 
[ Destination-Host ] 
* [ Class ] 
[ Origin-AAA-Protocol ] 
[ Origin-State-Id ] 
* [ Proxy-Info ] 
* [ Route-Record ] 
* [ AVP ]

If you have specs—make the most of them
Generate code, test input, spec-driven validation

Often, you’ll find that the spec is broken

(From rfc4005_nas.dia
Erlang/OTP’s Diameter application)
Trust/verify

Verify

Trust

Trust (assert) data from internal users

Check data from external users (specification-driven)
Information grows stale

Lifetime indicators?

Persistency

How long should data live?

“Unknown” is a useful indicator
Modeling data lifetimes

Don’t mix persistent and transient data

Persistency levels
- replicated disk
- replicated RAM
- replication factor

Erlang-style
- lightweight processes
- automatic GC
- single-assignment
- messaging

Transient request processes
The Where

In Emergency Response—obviously important

In tech, the Where can sometimes be inferred
But absence of signal is hard to interpret
Diagnosing absence of signal

“Virtual Device”

Information back-door

Status

(U DP)

Distributed Erlang

(TCP/IP)

© FEUERLABS 2013

Wednesday, 2 October 13
Knock-out Units

= The amount of service that can be lost in a crash

You **will** lose service—plan for it!

Better to fail distinctly than to pretend to function

**Invariants:** If they fail, all bets are off

![Diagram](Image)

Connection fan-out

Replication messages in-flight
Let it Crash.... or Try for a Result?

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 113569870251</td>
<td>7 113569870251</td>
</tr>
<tr>
<td>113569470251</td>
<td>113569470251</td>
</tr>
<tr>
<td>113669471251</td>
<td>113669581261</td>
</tr>
<tr>
<td>113669571251</td>
<td>113669581251</td>
</tr>
<tr>
<td>113669581261</td>
<td>113669581261</td>
</tr>
<tr>
<td>114669581262</td>
<td>114670581262</td>
</tr>
<tr>
<td>115670681262</td>
<td>115670681262</td>
</tr>
</tbody>
</table>

Tempting to always deliver a pretty result

A result that looks right, while erroneous, is often worse than no result at all
Conclusion

As programmers, we sometimes forget to model failure

Key is to think of information quality
- Data lifetime
- Data loss potential
- What data do I need for recovery?
- What failures can we discern?
- What interruptions are acceptable?
- What do our users expect?

Invariants
Questions?