Knowledge is Imperfect

Acting on Stale, Inconsistent or Missing Data

Ulf Wiger, Feuerlabs, Inc.

GOTO Aarhus 2013

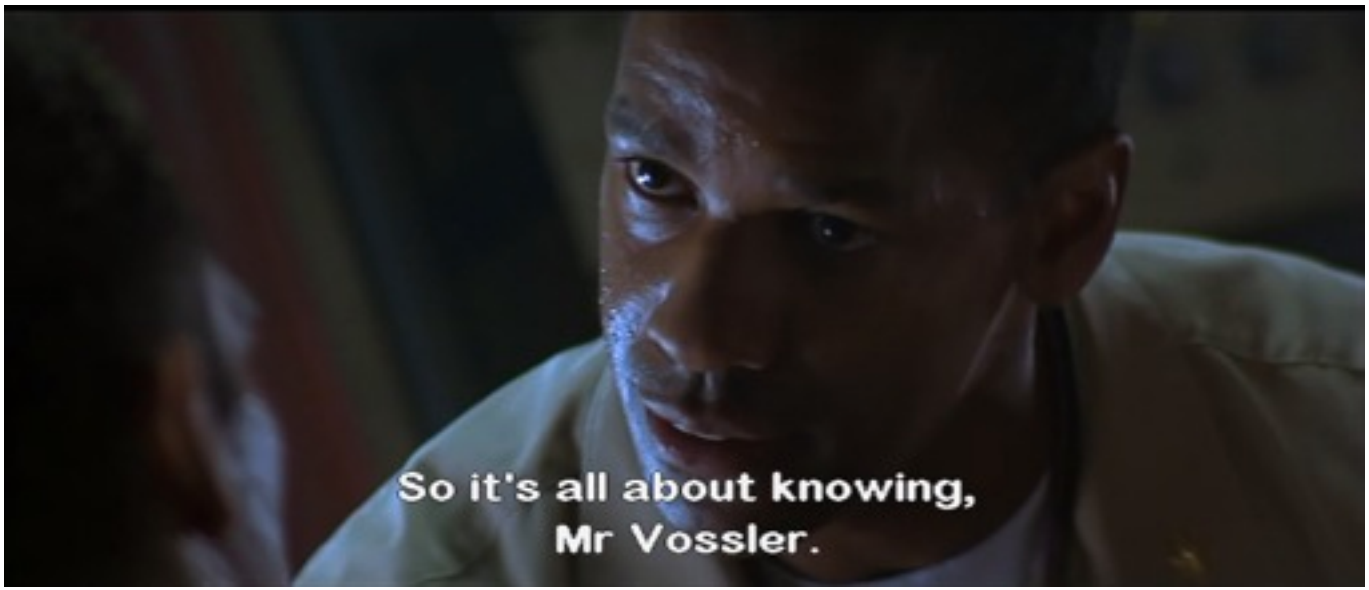
Outline

War stories

No code, no algorithms, no Hadoop

Thoughts

The code may well be broken before it's even written



Crimson Tide (1995) 1:21:26

Experience

Alaskan Adventure

Worked on military Command & Control and Emergency Response in Alaska 1989–1995

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

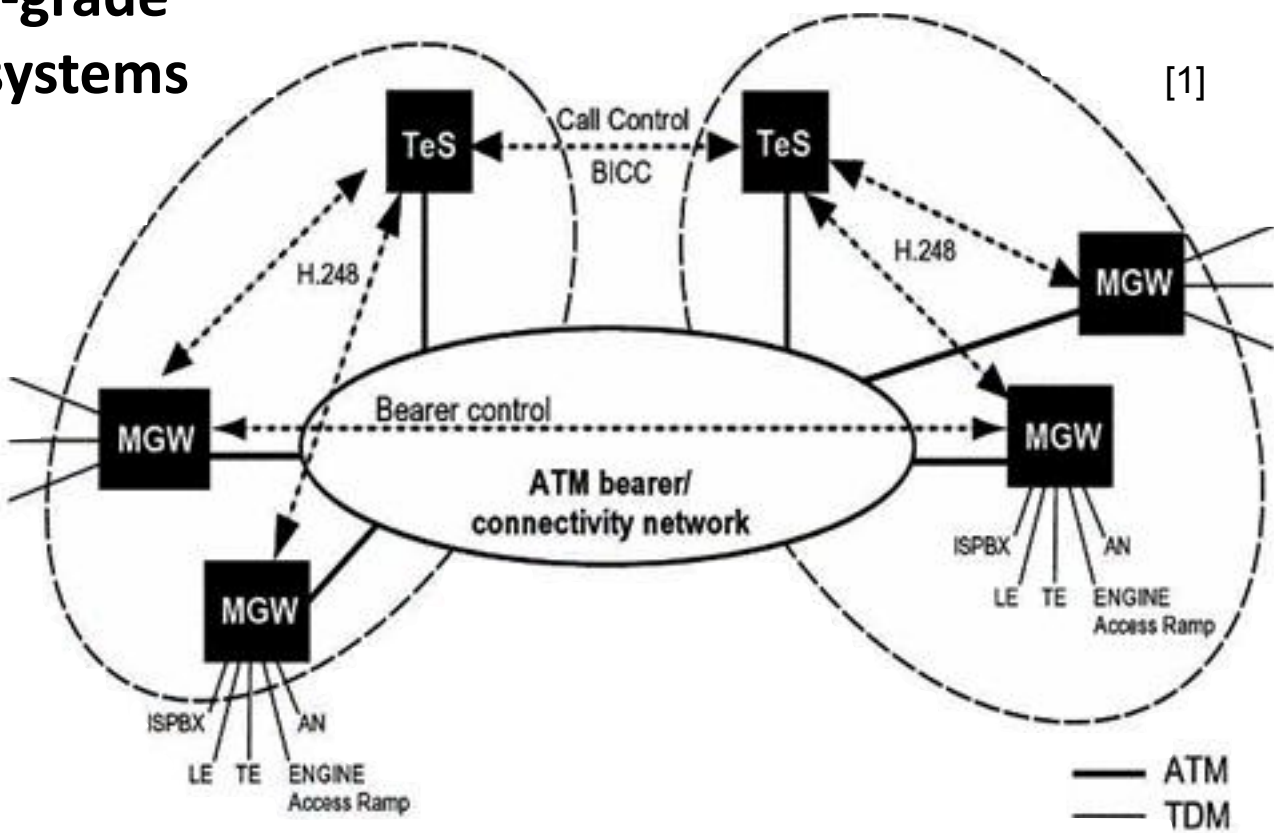
Pull information from any source

Ericsson adventure

13 years building telephony systems at Ericsson

World's first carrier-grade voice-over-packet systems

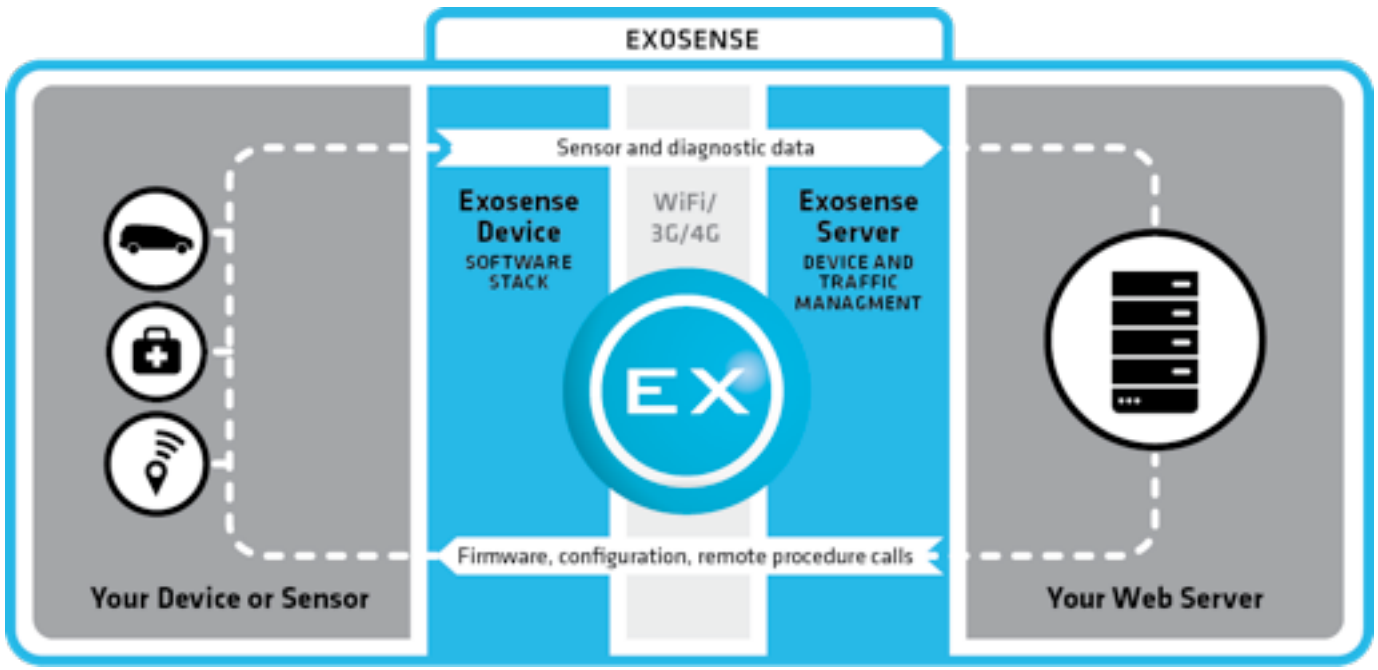
[1]



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!

US Marines CAC2 System

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

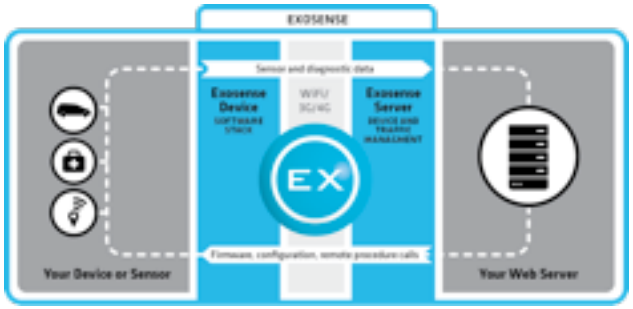
Few seconds setup time

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

Decision Support

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

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 ]
```

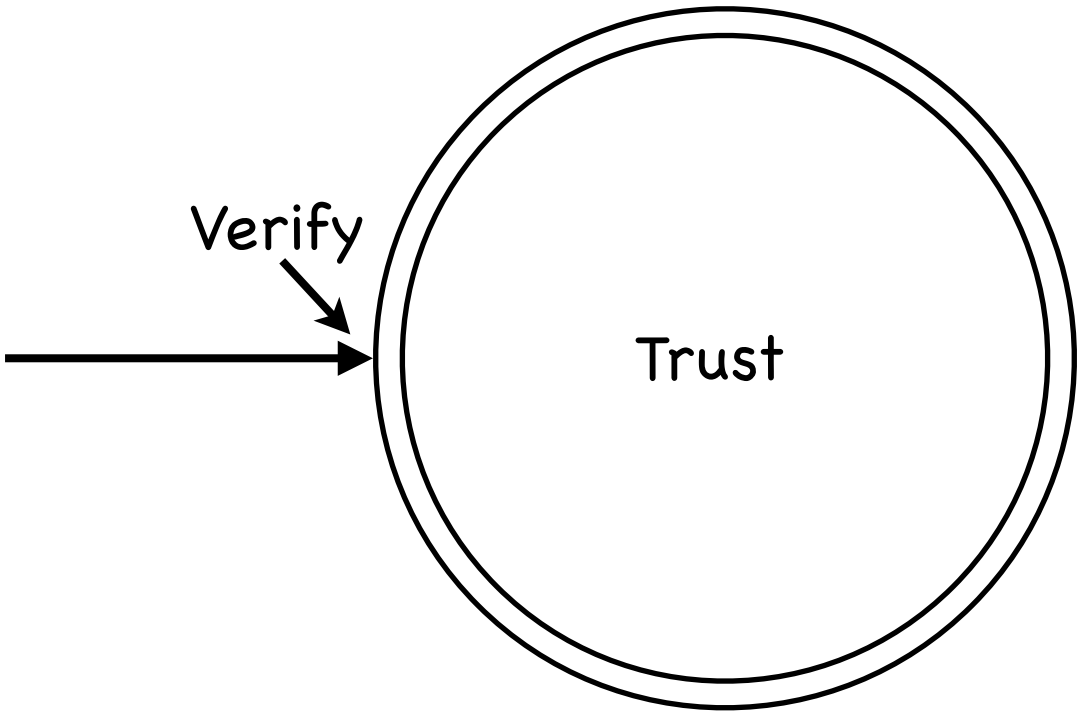
(From rfc4005_nas.dia
Erlang/OTP's Diameter application)

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

Trust/verify



Trust (assert) data from internal users

Check data from external users (specification-driven)

The When

Information grows stale

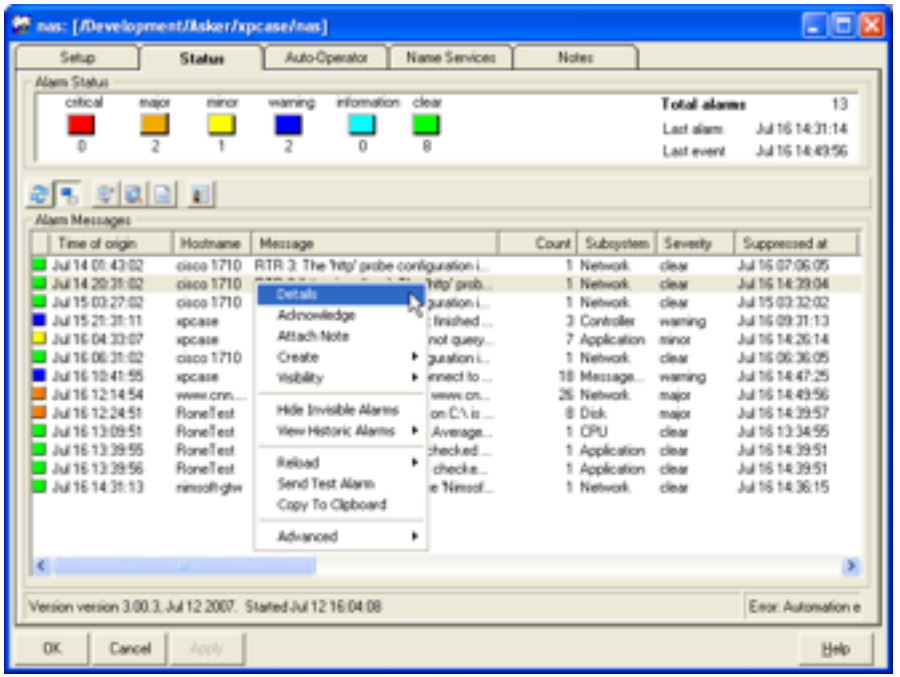
Lifetime indicators?

Persistency

How long should data live?

“Unknown” is a useful indicator

[2]



Modeling data lifetimes

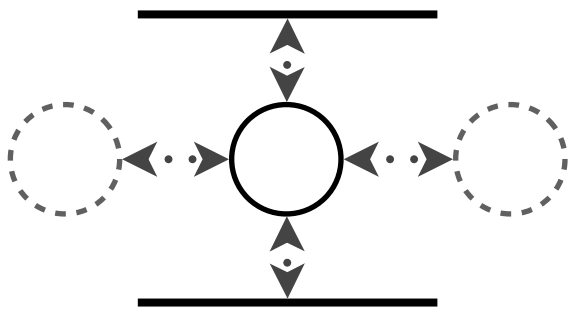
Don’t mix persistent and transient data

Persistency levels

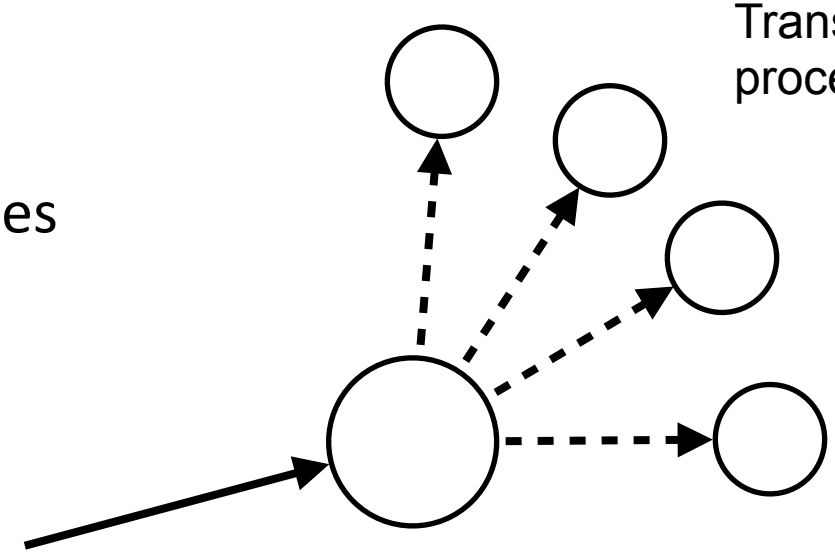
- replicated disk
- replicated RAM
- replication factor

Erlang-style

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



Transient request processes



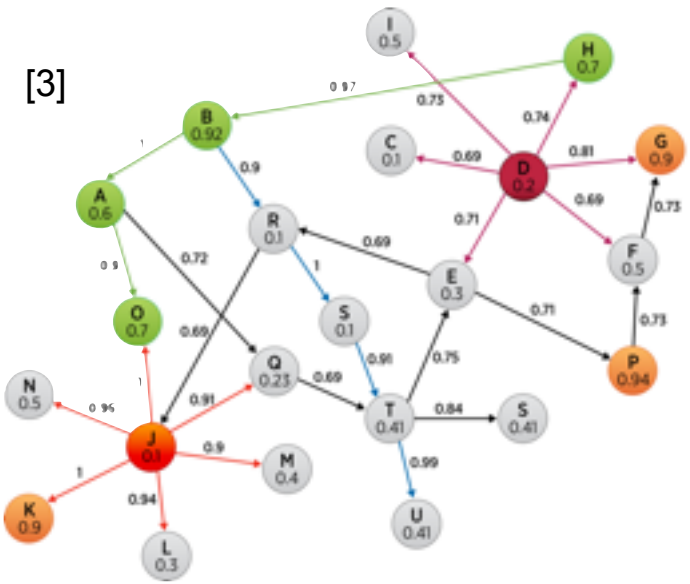
The Where

In Emergency Response—obviously important

In tech, the Where can sometimes be inferred

But absence of signal is hard to interpret

[3]

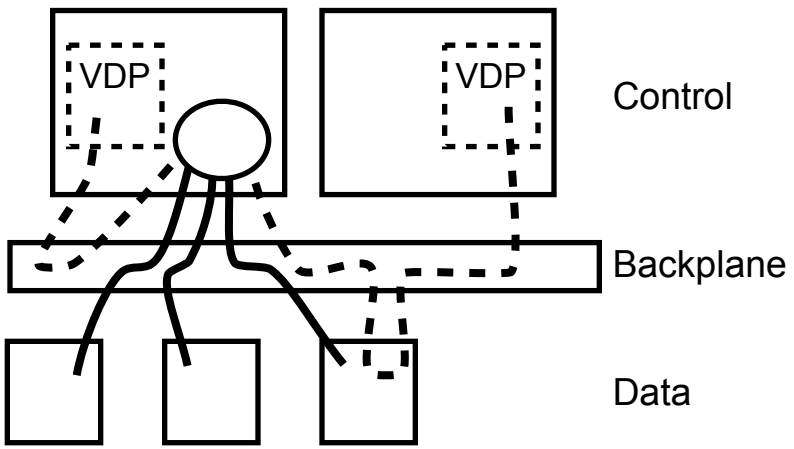


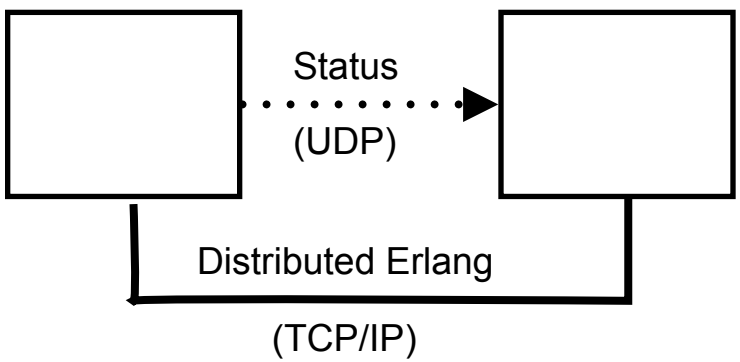
[4]

Diagnosing absence of signal

“Virtual Device”

Information back-door





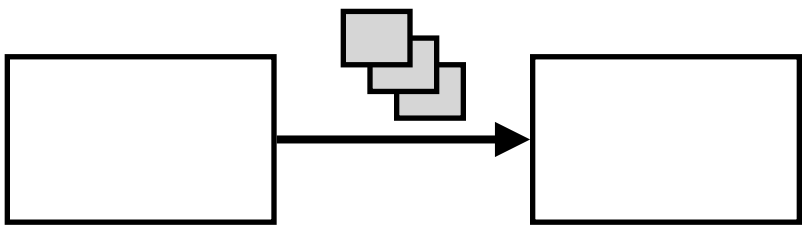
Knock-out Units

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

You will lose service—plan for it!

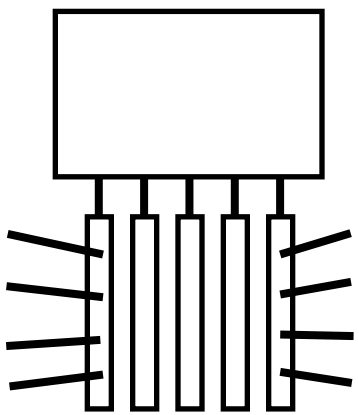
Better to fail distinctly than to pretend to function

Invariants: If they fail, all bets are off



Replication messages
in-flight

Connection fan-out



Let it Crash.... or Try for a Result?

Before	After
7 113569370251 11356937025	7 113569370251 11356937025
113569470251 113569470251	113569470251 113569470251
113669471251 113669471251	113869471251 113669471251
113669571251 113669581251	113669571251 113869581251
113669581261 113669581261	113669581261 113669581261
114669581262 114669581262	114669581262 114869581262
114670581262 114670581262	114670581262 114670581262
115670681262 115670681262	115670681262 115670681262

[5]

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?

[1] http://evaluation.nbu.bg/pub/NGN_MP_e_book_CD/DL_NGN_2004%20Module%205/Module%205/1.7%20softswithces.htm
[2] http://docs.nimsoft.com/prodhelp/en_US/Probes/Catalog/nas/3.6/index.htm?toc.htm?1942450.html
[3] http://labs.vmware.com/vmtj/an-anomaly-event-correlation-engine-identifying-root-causes-bottlenecks-and-black-swans-in-it-environments
[4] http://news.techeye.net/software/bespoke-os-blip-caused-chaos-in-the-air
[5] http://www.theregister.co.uk/2013/08/06/