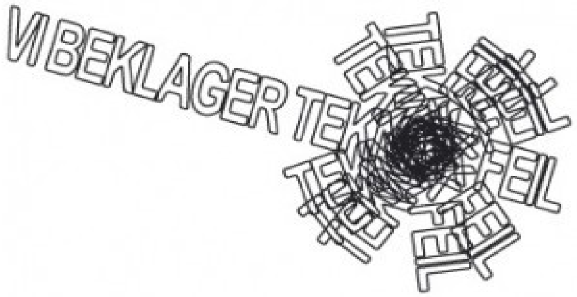
Manifesto for Agile Software Development
Customer collaboration
over contract negotiation
Individuals and interactions
over processes and tools
Working software
over comprehensive documentation
Responding to change
over following a plan

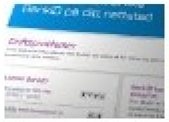
Do distribute!

The fallacies of Distributed Computing meets the Cloud...

Totto-11
totto@webstep.no
@javatotto/@tottoNOR



Mer it-trøbbel på Ahus
Lørdag kveld havarerte datasystemene på det nye sykehuset i Akershus.
Publisert:20.06.2011 kl 10:16

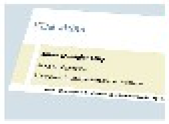


Bankid nede i seks timer
2,5 millioner nordmenn fikk ikke brukt Bankid, uvisst av hvilken grunn.
Publisert:06.07.2011 kl 13:31

Finanstilsynet slakter EDB Ergo
Refser selskapets tjenestekvalitet og risikoanalyse etter bankkrisen i påsken.
Publisert:20.06.2011 kl 15:10

Skandiabanken var nede - igjen
Nettbank og telefonlinjer døde. Kunder ble henvist til Facebook og Twitter.
Publisert:10.05.2011 kl 16:05

Ny bankkrasj for EDB Ergogroup
Betalingsterminaler og nettbanker var nede igjen. EDB Ergogroup beklager.
Publisert:28.04.2011 kl 16:10



Umulig å sjekke selvangivelsen
Også i år kneler Altinn-serverne under trykket fra nysgjerrige nordmenn.
Publisert:22.03.2011 kl 10:33

INTRO – BACKGROUND

Do distribute! (If not for scale so for availability..)

Totto

«To revolutionize the world, you have to win the big fights»

- Distributed systems geek
- Let the machines do the work..
- Tired of seeing systems fail..
- Been advocating and building *recilliant* systems for quite some time
- *"We can do **much** better"*
- *'It just makes sense'*

webstep

Distributed System

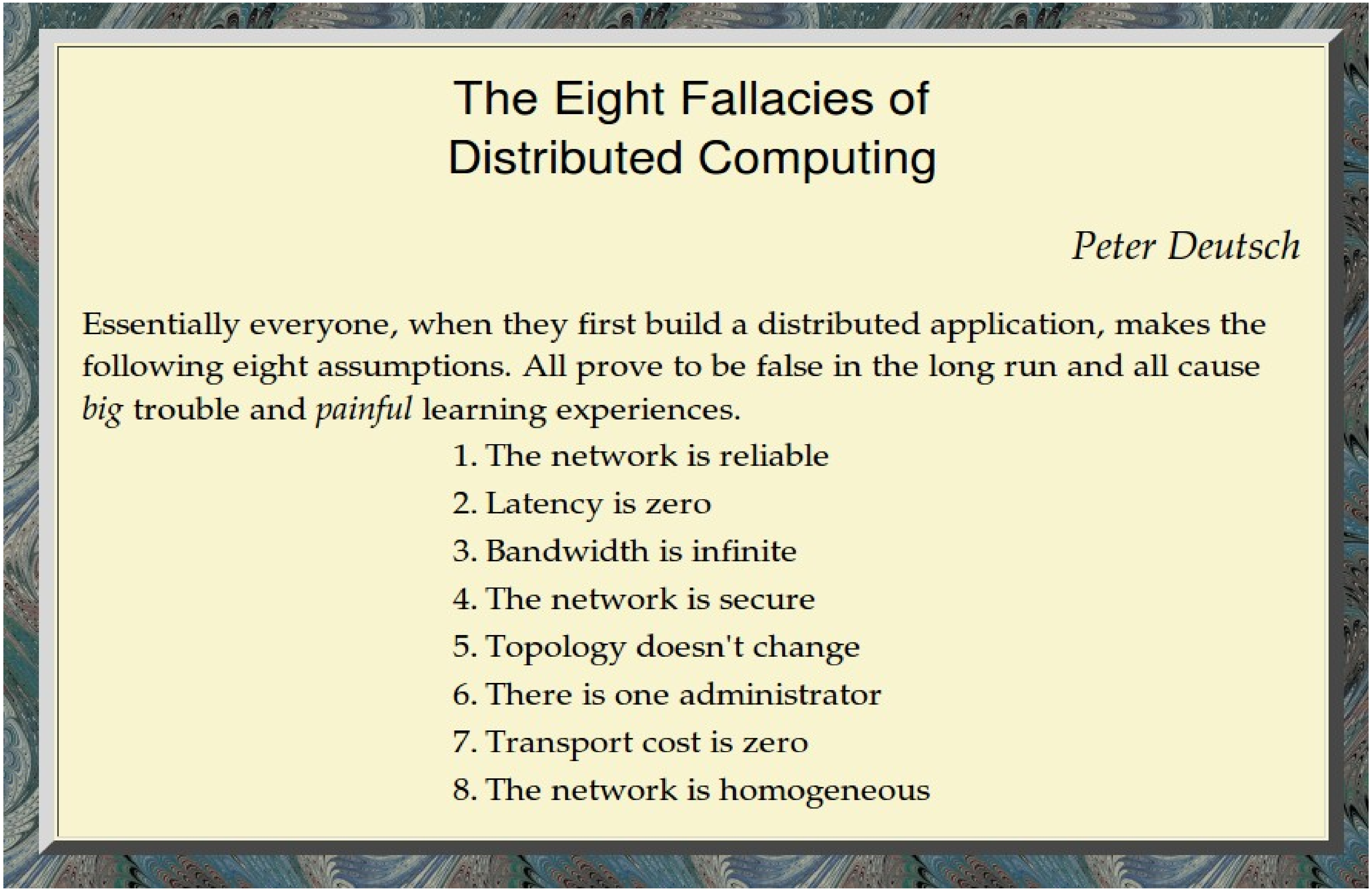
- A collection of independent computers that appears to its users as a single coherent system
  - - Tanenbaum and Steen: Distributed Systems: Principles and Paradigms

- **You know you have a distributed system when the crash of a computer you've never heard of stops you from getting any work done.**
  - - Leslie Lamport : Security Engineering: A Guide to Building Dependable Distributed Systems

- A distributed system is an application that executes a collection of protocols to coordinate the actions of multiple processes on a network, such that all components cooperate together to perform a single or small set of related tasks.
  - - Introduction to Distributed System Design – Google Code University

Distributed Computing

- is not GRID
    - Grid is **distributed SIMD**
- is not a Cluster
    - Cluster looks like **One Machine**
        - "Failure does not exist"
- Is nothing like *plain* J2EE - **think different**!



The Eight Fallacies of Distributed Computing

Peter Deutsch

Essentially everyone, when they first build a distributed application, makes the following eight assumptions. All prove to be false in the long run and all cause *big* trouble and *painful* learning experiences.

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

Background

- The Fallacies were coined 17 years ago, but they apply just as well today.

- *Back then folks might have been fooled into actually believing them, but we're smarter than that,* ***right****?*

- Knowing what gotchas await our services in the cloud, can help us build better, resilient, faster and more performant applications.

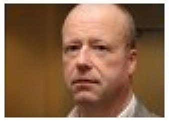
- Isn't that what we all want?

Initial refections on developers

- The two types of developers:
  - application developer
  - systems developer

- The world of application developers
  - driven by: $$$ == Business features
    (read: single thread, “perfect world”, “solved in infrastructure”)
  - 2011: RIP (examples: Altinn, NAV)

- The world of systems developers
  - driven by: Street-cred (coolest tech)
    (read: manual thread spawning, network=retry et all)
  - 2011: Dead by double errors
    (examples: Telenor, EDBErgogroup)



Nettverksproblemer bak Ahus-kollapsen
En svitsj får skylden for at hele ip-nettet til Ahus gikk ned for telling i helgen.
Publisert:20.06.2011 kl 14:36

1. The network is reliable

- As a developers we have a number of useful abstractions available for working with network requests. Unfortunately none of them can actually spare us from needing to consider the realities of an unreliable network. And since the abstractions try to hide the unreliability most developers tend to forget to handle unreliable situations resulting in component and system failure.

```
try {
        URL url = new URL( args[0] );
        BufferedReader in = new BufferedReader(new
                                        InputStreamReader(url.openStream()));
        String str;
        while ((str = in.readLine()) != null)  {
          System.out.println(str);
        }
        in.close();
    }
    catch (MalformedURLException e) {}
    catch (IOException e) {}
}
```

Altinn er nede i leveringsrushet

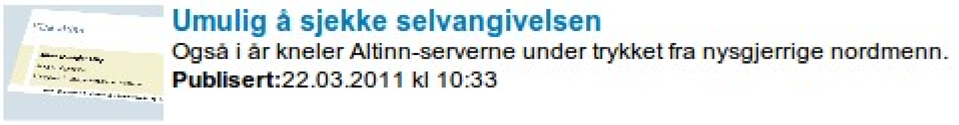
Næringsdrivende raser etter fire dagers nedetid før leveringsfristen.

Publisert:14.06.2010 kl 11:46

2. Latency is zero

- Most network API and wrappers try to hide the fact that there can be latency in the network. I guess most of the developers in this room have experienced some kind of system failure being caused by sudden high increase in latency. If the system design does not take this law into account, you are certain to be bitten but such consequences.

- Number of requests, size of message, lazy loading, movable code is techniques to consider..



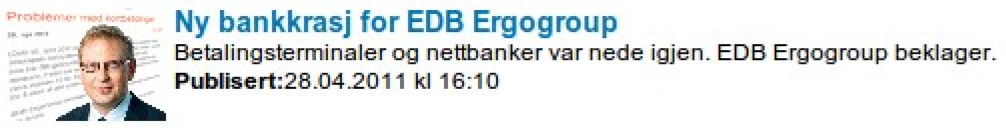
Umulig å sjekke selvangivelsen

Også i år kneler Altinn-serverne under trykket fra nysgjerrige nordmenn.

Publisert:22.03.2011 kl 10:33

3. Bandwidth is infinite

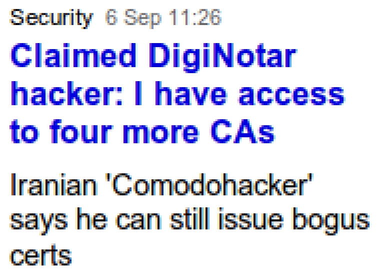
- Bandwidth has never been as high as it is in 2011. But as the recent Telenor Mobil failure, the “askesky” problems and the yearly Altinn failures keeps reminding us on, high! =infinite. (you might add DoS/DDoS to that list).
- If the system design does not account for this failure, spectacular crashes and service unavailabilities is always the result.


Ny bankkrasj for EDB Ergogroup

Betalingsterminaler og nettbanker var nede igjen. EDB Ergogroup beklager.

Publisert:28.04.2011 kl 16:10

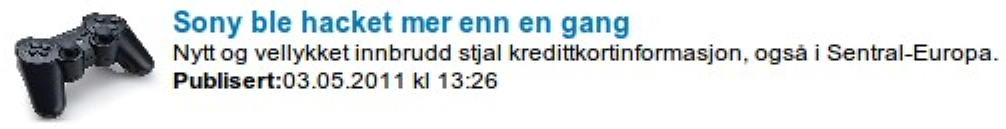
4. The network is secure


Security 6 Sep 11:26

Claimed DigiNotar hacker: I have access to four more CAs

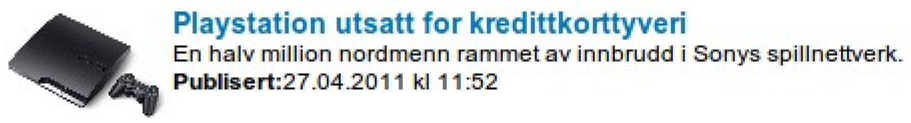
Iranian 'Comodohacker' says he can still issue bogus certs

- Statistics published at Aladdin.com shows that:
  - "**For 52% of the networks the perimeter is the only defense**
- Through the continual 24x7 monitoring of hundreds of Fortune 1000 companies, RipTech has discovered several extremely relevant trends in information security. Among them:
  - General Internet attack trends are showing a 64% annual rate of growth
  - The average company experienced 32 attacks per week over the past 6 months
  - Attacks during weekdays increased in the past 6 months"
- The implications of network (in)security are obvious
  - **you need to build security into your solutions from Day 1.**
  - which 9 of 10 projects today still fail to do..


Sony ble hacket mer enn en gang

Nytt og vellykket innbrudd stjal kredittkortinformasjon, også i Sentral-Europa.

Publisert:03.05.2011 kl 13:26


Playstation utsatt for kredittkorttyveri

En halv million nordmenn rammet av innbrudd i Sonys spillnettverk.

Publisert:27.04.2011 kl 11:52

Feil i programvare knelte Telenor
Telenor har funnet feilen som gjorde at selskapets mobilnett gikk ned i juni.
Publisert:19.07.2011 kl 13:03

5. Topology does not change

- That's right, it doesn’t--as long **as it stays in the test lab.**
- When you deploy an application in the wild (that is, to an organization), the network topology is usually out of your control. The operations team (IT) is likely to add and remove servers every once in a while and/or make other changes to the network ("this is the new Active Directory we will use for SSO ; we're replacing RIP with OSPF and this application's servers are moving into area 51" and so on).
- Lastly there are server and network faults which can cause routing changes. When you're talking about clients, the situation is even worse. There are laptops coming and going, wireless ad-hoc networks , new mobile devices. **In short, topology is changing constantly**

Finanstilsynet slakter EDB Ergo

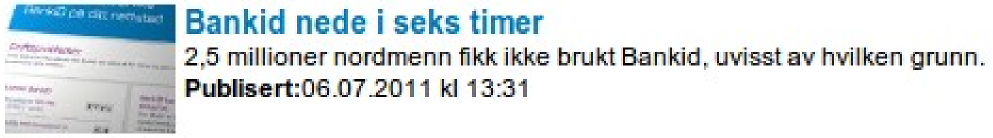
Refser selskapets tjenestekvalitet og risikoanalyse etter bankkrisen i påsken.

Publisert:20.06.2011 kl 15:10

6. There is one administrator

- At this point you may say "Okay, there is more than one administrator. But why should I care?" Well, as long as everything works, maybe you don't care. You do care, however, when things go astray and there is a need to pinpoint a problem (and solve it).

- **The famous developers versus operations battle**


Bankid nede i seks timer

2,5 millioner nordmenn fikk ikke brukt Bankid, uvisst av hvilken grunn.

Publisert:06.07.2011 kl 13:31

7. Transport cost is zero

- The costs (as in cash money) for setting and running the network are free. This is also far from being true. There are costs--costs for buying the routers, costs for securing the network, costs for leasing the bandwidth for Internet connections, and costs for operating and maintaining the network running. Someone, somewhere will have to pick the tab and pay these costs.

- Imagine you have successfully built a Google-killer search engine but you will fail if you neglect to take into account the costs that are needed to keep your service up, running, and responsive (E3 Lines, datacenters with switches, SANs etc.).

- The takeaway is that even in situations you think the other fallacies are not relevant to your situation because you rely on existing solutions

Skandiabanken var nede - igjen

Nettbank og telefonlinjer døde. Kunder ble henvist til Facebook og Twitter.

Publisert:10.05.2011 kl 16:05

8. The network is homogeneous

- This fallacy was added to the original seven by James Gosling, creator of Java, in 1997.
- It is worthwhile to pay attention to the fact the network is not homogeneous at the application level. The implication of this is that you have to assume interoperability will be needed sooner or later and be ready to support it from day one (or at least design where you'd add it later).
- **Do not rely on proprietary protocols**--it would be harder to integrate them later. Do use standard technologies that are widely accepted; the most notable examples being JSON, XML or Web Services. By the way, much of the popularity of XML and Web Services can be attributed to the fact that both these technologies help alleviate the affects of the heterogeneity of the enterprise environment.

A few word on distributed systems

- Distributed systems must deal with:
  - Failure of part of the system
  - Accumulation of outdated and unwanted information
- Network:
  - resilliant (motstandsdyktighet)
    - Fallacies: 1, 4, 5, 6, 8
  - discovery (fleksibilitet)
    - Fallacies: 1, (2), (3), 4, 5, 6, (7), 8
- ***Simplified:***
  - discovery
  - lease (and extensions like mobile code)

A quick walk through the memory lane

Lets see how far we have come the last 15+ years..(and reflect of how we got where we are..)

A few words on distributed systems the last 20 years

- 1994: TCP/IP wrappers
  - client-server, object models, memory leaks, homemade protocols
- 1998: (Com+/EJB)
  - the year of containers. Application developers should write applications in server containers.
- 2000: Internet (IFrame)
  - mostly mashups, frontend developers mix&match content
- 2008: REST
  - resources (data) *returned* to the internet
- 2011: The Cloud
  - Infrastructure as a Service. Hardware just another “software” service. The end of the SLA age..

Rest || Web

REST is an architectural style which has 4 fundamental constraints:

* Identification of **resources**
* Manipulation of **resources** through representations
* Self-descriptive messages
* Hypermedia as the engine of application state

2008 – Let us have a look at the fallacies in the WEB/REST age

The year data/resources returned to the Internet

REF: Tim Bray: The Web vs. the Fallacies- 2009.

webstep

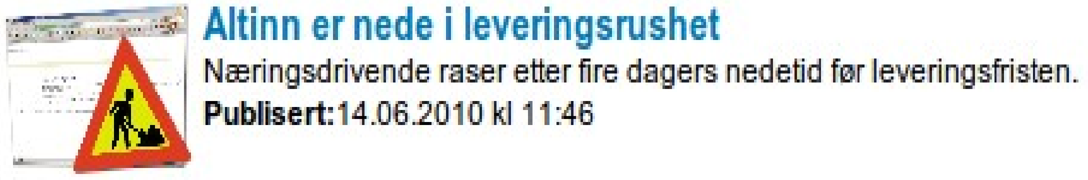
Nettverksproblemer bak Ahus-kollapsen

En svitsj får skylden for at hele ip-nettet til Ahus gikk ned for telling i helgen.

Publisert:20.06.2011 kl 14:36

1. The network is reliable

- HTTP helps here in a couple of different ways. Most obviously, connections are brief; I've never seen much in the way of measurements, but I'd expect the average connection lifetime to be under a second. Compared to a traditional networked system with connections whose lifetime approximates that of the application, this moves the likelihood of experiencing a damaging connection breakdown while application code is running from 'essentially always' to 'rather rarely'.
- Second, the clarity about GET, PUT, and DELETE being idempotent, while POST isn't, helps hugely. Most obviously, if a GET gets a network blowup, just do it again. And if the breakage hits a POST, well, it probably only hits one, and this places very clear boundaries around the repair and recovery that an app needs to handle.

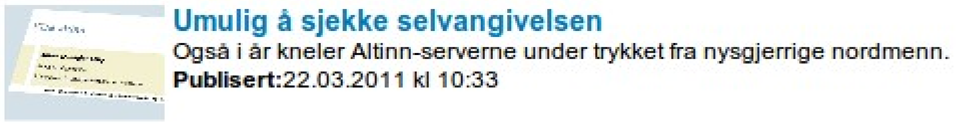

Altinn er nede i leveringsrushet

Næringsdrivende raser etter fire dagers nedetid før leveringsfristen.

Publisert:14.06.2010 kl 11:46

2. Latency is zero

- The Web actually makes the latency problem worse, because every interchange, on average, requires connection setup/teardown.
- Since it's not realistic to expect anything like keystroke-level latency across the Net, the correct engineering solution is to defuse the expectation.


Umulig å sjekke selvangivelsen
Også i år kneler Altinn-serverne under trykket fra nysgjerrige nordmenn.
Publisert:22.03.2011 kl 10:33

3. Bandwidth is infinite

- Here the Web has been a wonderful teacher of networking realities to the non-technical.
- Time after time, you'll see messages, between computing civilians, of the form 'Sorry that this picture is so big' because they know perfectly well that it's going to slow down the experience of seeing it.

4. The network is secure

Slik svindles norske bankkunder

DnB NORs kunder har vært utsatt for fire trojanerangrep i år. Her kan du lese om hvordan angrepene artet seg, og hvordan de ble håndtert.

Security 6 Sep 11:26

Claimed DigiNotar hacker: I have access to four more CAs

Iranian 'Comodohacker' says he can still issue bogus certs

- This is probably the fallacy least-well-addressed by the Web. True, people have become more aware that There Are Bad Guys out there, and they need to be careful. But not nearly enough.

- Also, let's grant that TLS, properly deployed, has been pretty well Good Enough to run apps in a mostly-secure way in a hostile environment. But who among us would be surprised if someone turned up a catastrophic flaw, perhaps not in TLS itself, but in one or two widely-deployed implementations? Who's to say that someone hasn't, already?

- Anyhow, the Web technologies mean that application builders can survive even while subject to one or more of The Fallacies. But not this one.

Feil i programvare knelte Telenor
Telenor har funnet feilen som gjorde at selskapets mobilnett gikk ned i juni.
Publisert:19.07.2011 kl 13:03

5. Topology does not change

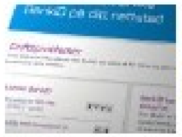
- By making almost all our apps Web-based, and thus having everyone address everything with URIs, we all agree to share solutions to routing and addressing problems; solutions provided by the DNS, the network stack, and the Internet backbone operators.
- This doesn't mean the solutions are easy or cheap or perfect; it just means that application builders almost never have to think about the problem.

Finanstilsynet slakter EDB Ergo
Refser selskapets tjenestekvalitet og risikoanalyse etter bankkrisen i påsken.
Publisert:20.06.2011 kl 15:10

6. There is one administrator

- Well yeah, there isn't. But who cares, any more? Web architecture makes addressing decentralized. Thus when an administrator screws up, or imposes policies that seem good to him or her and insane to you, the damage is limited to that person's URI space.

- Also, Web architecture, which requires that you talk about things in terms of the URIs you use to address them and the data formats you use to transmit them, makes it a whole lot easier to achieve administrative coherence even when there are millions of administrators.



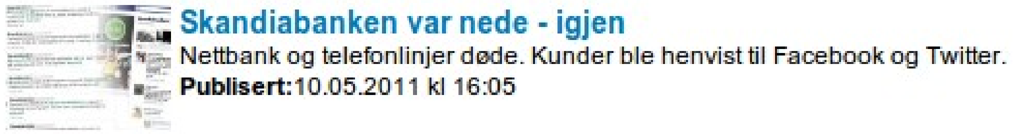
Bankid nede i seks timer

2,5 millioner nordmenn fikk ikke brukt Bankid, uvisst av hvilken grunn.

Publisert:06.07.2011 kl 13:31

7. Transport cost is zero

- Once again, the Web has been a wonderful teacher of networking realities to the non-technical. Time after time, you'll see messages, between computing civilians, of the form 'Sorry that this picture is so big' because they know perfectly well that it's going to slow down the experience of seeing it.


Skandiabanken var nede - igjen
Nettbank og telefonlinjer døde. Kunder ble henvist til Facebook og Twitter.
Publisert:10.05.2011 kl 16:05

8. The network is homogeneous

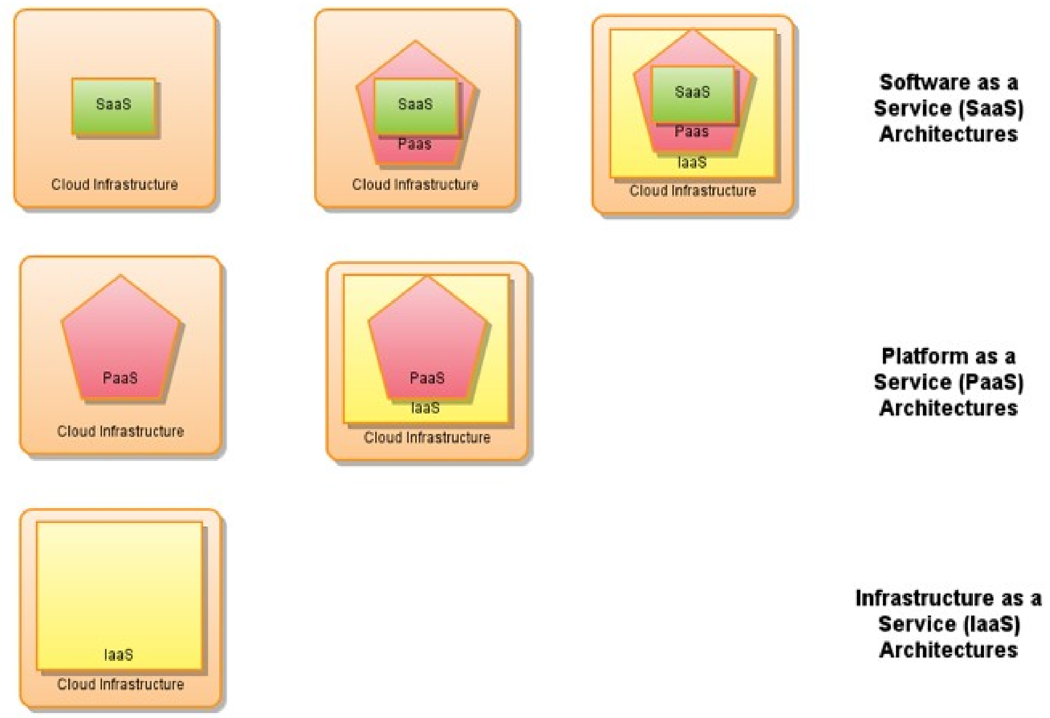
- This is perhaps the Web's single greatest triumph. For decades we thought we could extend object models and APIs and lots of other programming concepts over the network. This was a fool's errand, because all you can do with a network is send messages over it.

- The Web doesn't do APIs and object models, it's just a set of agreements over what messages you're going to send and what messages expect back in return. Which, as a side-effect, makes heterogeneity a non-issue.

Fallacies and the Web/REST - recap

* The web has taught us some of the fallacies

* Security and reliability are still major flaws



Q: What's the difference between vitalization and Cloud?

A: Ask for 200 powerful workstations for a month. Compare costs and delivery time..

2011 – The Cloud

The year hardware stopped being different than any other Internet software service

REF: Brian Doll: The Fallacies of Distributed Computing Reborn: The Cloud Era

Fallacies and the Cloud, some observations...

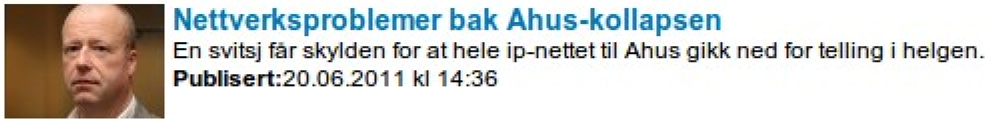
"I knew to expect higher rates of individual instance failure in AWS, but I hadn't thought through some of these sorts of implications."

"AWS networking has more **variable latency**. We've had to be much more structured about "over the wire" interactions, even as we've transitioned to a more highly distributed architecture."

"Co-tenancy can introduce **variance in throughput** at any level of the stack. You've got to either be willing to abandon any specific subtask, or manage your resources within AWS to avoid co-tenancy where you must."

"We're designing each distributed system to **expect and tolerate failure** from other systems on which it depends."

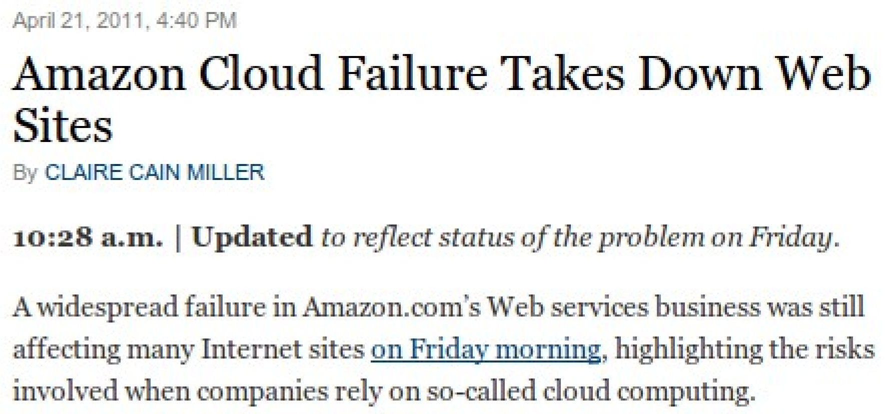
– John Ciancutti, Vice President of Personalization Technology at Netflix, Inc


Nettverksproblemer bak Ahus-kollapsen

En svitsj får skylden for at hele ip-nettet til Ahus gikk ned for telling i helgen.

Publisert:20.06.2011 kl 14:36

1. The network is reliable


April 21, 2011, 4:40 PM

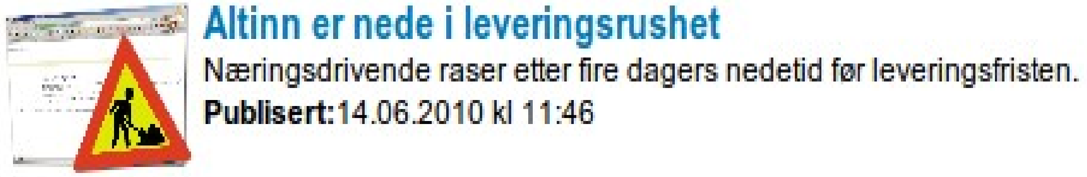
Amazon Cloud Failure Takes Down Web Sites

By CLAIRE CAIN MILLER

10:28 a.m. | **Updated** *to reflect status of the problem on Friday.*

A widespread failure in Amazon.com's Web services business was still affecting many Internet sites on Friday morning, highlighting the risks involved when companies rely on so-called cloud computing.

- Design your system as a resilient distributed system...
- The network is not reliable. When you add up all the points of failure for all the various services and APIs you use, the odds of failure are not just high, they are a given. Instead, **build an app that can function at reduced capacity when a given service is offline.**


Altinn er nede i leveringsrushet

Næringsdrivende raser etter fire dagers nedetid før leveringsfristen.

Publisert:14.06.2010 kl 11:46

2. Latency is zero

- The latency in the cloud is higher (often much higher) than in the traditional data-center. Latency will also vary much more in the cloud, where noisy neighbors is a fact of life.
- **Ignore latency online and you're likely throwing away customers** from half the globe.
- Consider moving the service closer to the customer using CDN and multi-zone deployment

Umulig å sjekke selvangivelsen

Også i år kneler Altinn-serverne under trykket fra nysgjerrige nordmenn.

Publisert:22.03.2011 kl 10:33

3. Bandwidth is infinite

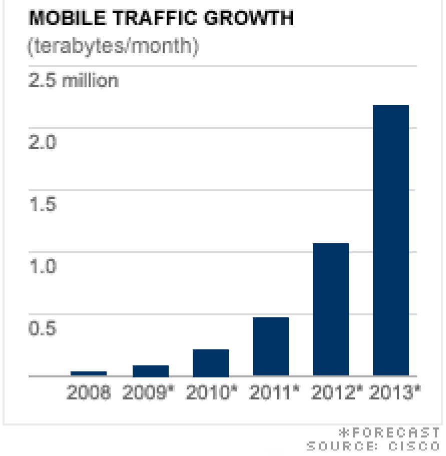
- Nowadays *everyone* has high-speed Internet access at home.

- A few years ago something started to change.
  - Web applications were being used not just over high-speed Internet connections, but over phone networks.
  - Using web apps on mobile devices and tablets reminds us that the **bandwidth problems did not disappear with the dial-up modem**

- Stay mindful of how much data you're shipping across the wire and all your users will be grateful.

On the move

Mobile data traffic, measured in terabytes per month, will keep growing.



Ny bankkrasj for EDB Ergogroup
Betalingsterminaler og nettbanker var nede igjen. EDB Ergogroup beklager.
Publisert:28.04.2011 kl 16:10

4. The network is secure

- The cloud is a good teacher, as the “perimeter security” excuse is **dead.**

- **Firesheep** brought some necessary attention last year to the prevalence of private security tokens littering insecure connections.

- The network is not secure, and the more we move our lives online the higher the risk of exposure.

- **Keep a security mindset when developing your apps and keep your users safe.**

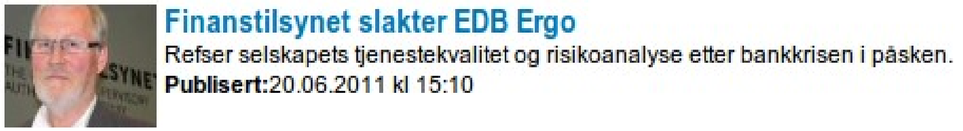
Feil i programvare knelte Telenor
Telenor har funnet feilen som gjorde at selskapets mobilnett gikk ned i juni.
Publisert:19.07.2011 kl 13:03

5. Topology does not change

- One of the biggest benefits of moving applications to the cloud is the **ability to change topology at will.**

- Upgrade the CPU on your database server in the middle of the day. Add reverse proxy servers, cache servers, change CDN providers, migrate availability zones.

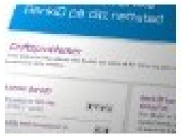
- Topology changes all the time. **Depending on a static infrastructure design not only limits your ability to respond to change, it increases the likelihood of site-wide outages.**



Finanstilsynet slakter EDB Ergo
Refser selskapets tjenestekvalitet og risikoanalyse etter bankkrisen i påsken.
Publisert:20.06.2011 kl 15:10

6. There is one administrator

- Of course there isn’t just one administrator. Even with applications hosted in your own private data-center, your applications are likely interacting with systems outside your administrative control.
- These systems may have performance, availability or security issues that you have no direct influence over.
- Staying mindful that these systems are beyond your control can help you **ensure they have minimal impact on your services** when they are unresponsive.



Bankid nede i seks timer

2,5 millioner nordmenn fikk ikke brukt Bankid, uvisst av hvilken grunn.

Publisert:06.07.2011 kl 13:31

7. Transport cost is zero

- The cloud computing business models are a great tutor in learning that transport cost is not frree.

- Not only is transport cost not zero, it’s priced like any other commodity and can be purchased per transaction, per gigabyte, per compute hour, etc.

- Cloud storage costs (and transport of that storage) are a major component of application hosting costs, just ask anyone doing video streaming online how close to zero their transport costs are.

Skandiabanken var nede - igjen

Nettbank og telefonlinjer døde. Kunder ble henvist til Facebook og Twitter.

Publisert:10.05.2011 kl 16:05

8. The network is homogeneous

- The best part of this fallacy is that it might actually be true. Almost. Between REST and JSON APIs, have you ever had to think about the implementation of an external service, beyond satisfying your curiosity?

- You’re much more likely to see libraries producing invalid messages than you are operating systems playing a role.

Fallacies and the cloud, recap

- As the cloud pushes our architectures in the distributed system direction, the fallacies are now more important than ever
- We can use the cloud providers SaaS and PaaS products to help us create resilient systems
- Security is not optional
-
- Perimeter defense is dead in 2011
-
- “Service Level Agreements (SLA)” in the cloud means building distributed systems on the cloud platform (SLA handled in software)

Some Cloud Design Blueprints

Some examples of how to pick smart from Cloud offerings without breaking the fallacies of distributed computing

webstep

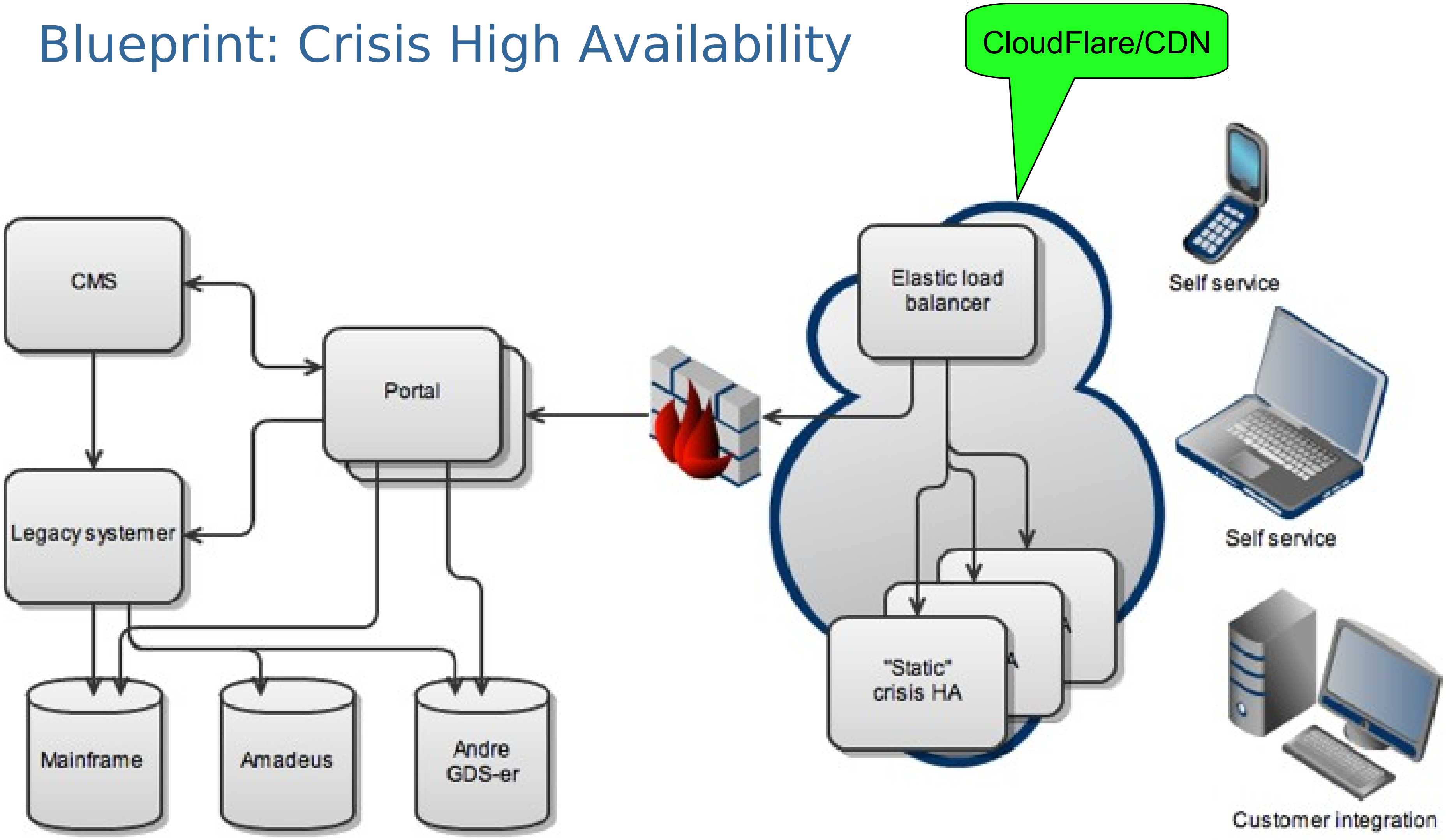
Chaos Monkey

- We've sometimes referred to the Netflix software architecture in AWS as our Rambo Architecture. Each system has to be able to succeed, no matter what, even all on its own. We're designing each distributed system to **expect and tolerate failure** from other systems on which it depends.

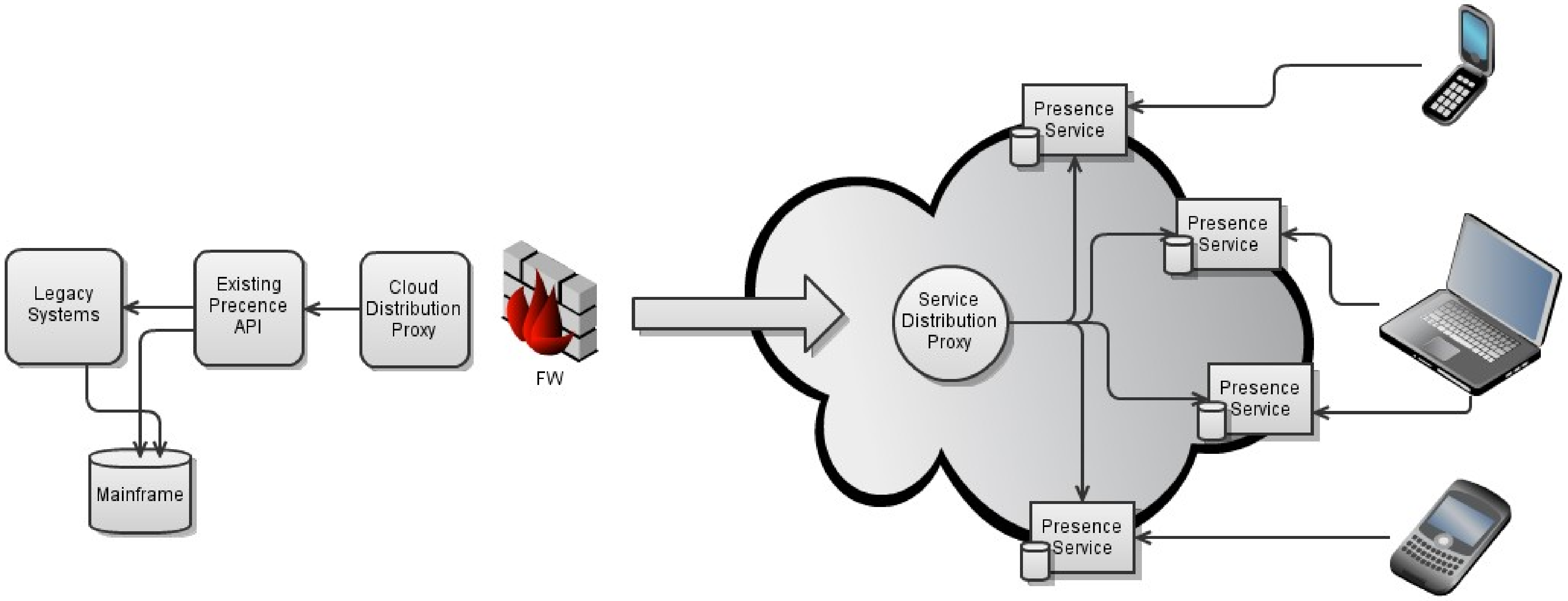
- If our recommendations system is down, we degrade the quality of our responses to our customers, but we still respond. We'll show popular titles instead of personalized picks. If our search system is intolerably slow, streaming should still work perfectly fine.

- One of the first systems our engineers built in AWS is called the Chaos Monkey. **The Chaos Monkey's job is to randomly kill instances and services within our architecture.** If we aren't constantly testing our ability to succeed despite failure, then it isn't likely to work when it matters most – in the event of an unexpected outage.

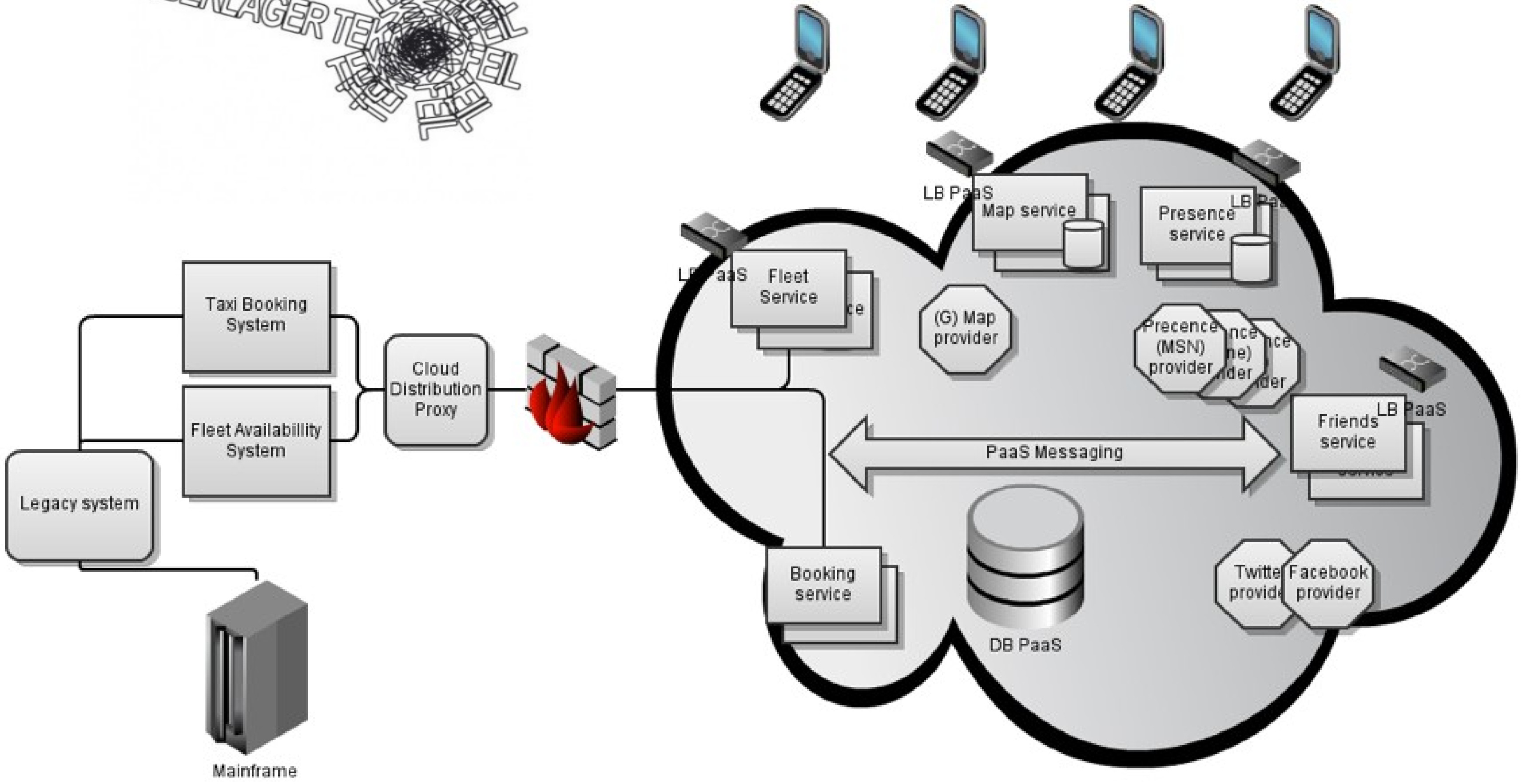
Blueprint: Crisis High Availability



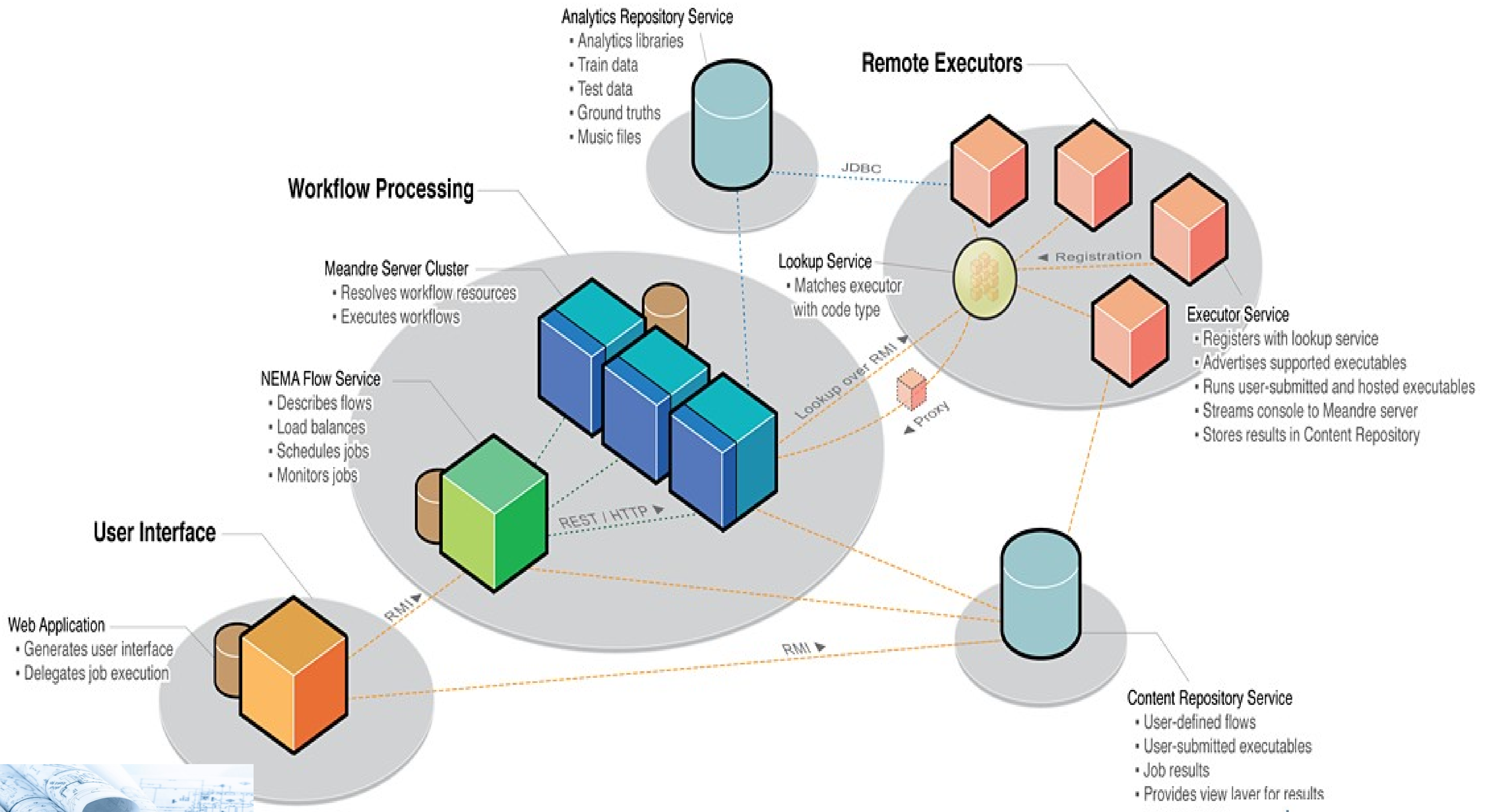
Blueprint: Cloud Distribution



Enterprise Architecture in the Cloud



A real distributed system...


Analytics Repository Service
• Analytics libraries
• Train data
• Test data
• Ground truths
• Music files
Remote Executors
JDBC
Workflow Processing
Meandre Server Cluster
• Resolves workflow resources
• Executes workflows
Lookup Service
• Matches executor with code type
Registration
Executor Service
• Registers with lookup service
• Advertises supported executables
• Runs user-submitted and hosted executables
• Streams console to Meandre server
• Stores results in Content Repository
Lookup over RMI
Proxy
NEMA Flow Service
• Describes flows
• Load balances
• Schedules jobs
• Monitors jobs
REST / HTTP
User Interface
RMI
Web Application
• Generates user interface
• Delegates job execution
RMI
Content Repository Service
• User-defined flows
• User-submitted executables
• Job results
• Provides view layer for results

webstep

Where to go next?

Some literature, links and open source to get you going in the right direction.

Some links

- http://techblog.netflix.com/2010/12/5-lessons-weve-learned-using-aws.html
- www.rgoarchitects.com/Files/fallacies.pdf
- http://blog.newrelic.com/2011/01/06/the-fallacies-of-distributed-computing-reborn-the-cloud-era/
- REST in Practise
- wiki.cantara.no

Q&A

This is today's last session, so we'll discuss as long as you please before continuing the discussion at ***AweZone****.*