Software Architecture

Henrik Møller Hansen

Agenda

- About me and my work
- Some general thoughts on software architecture
- Case walkthrough – Flight plan filing

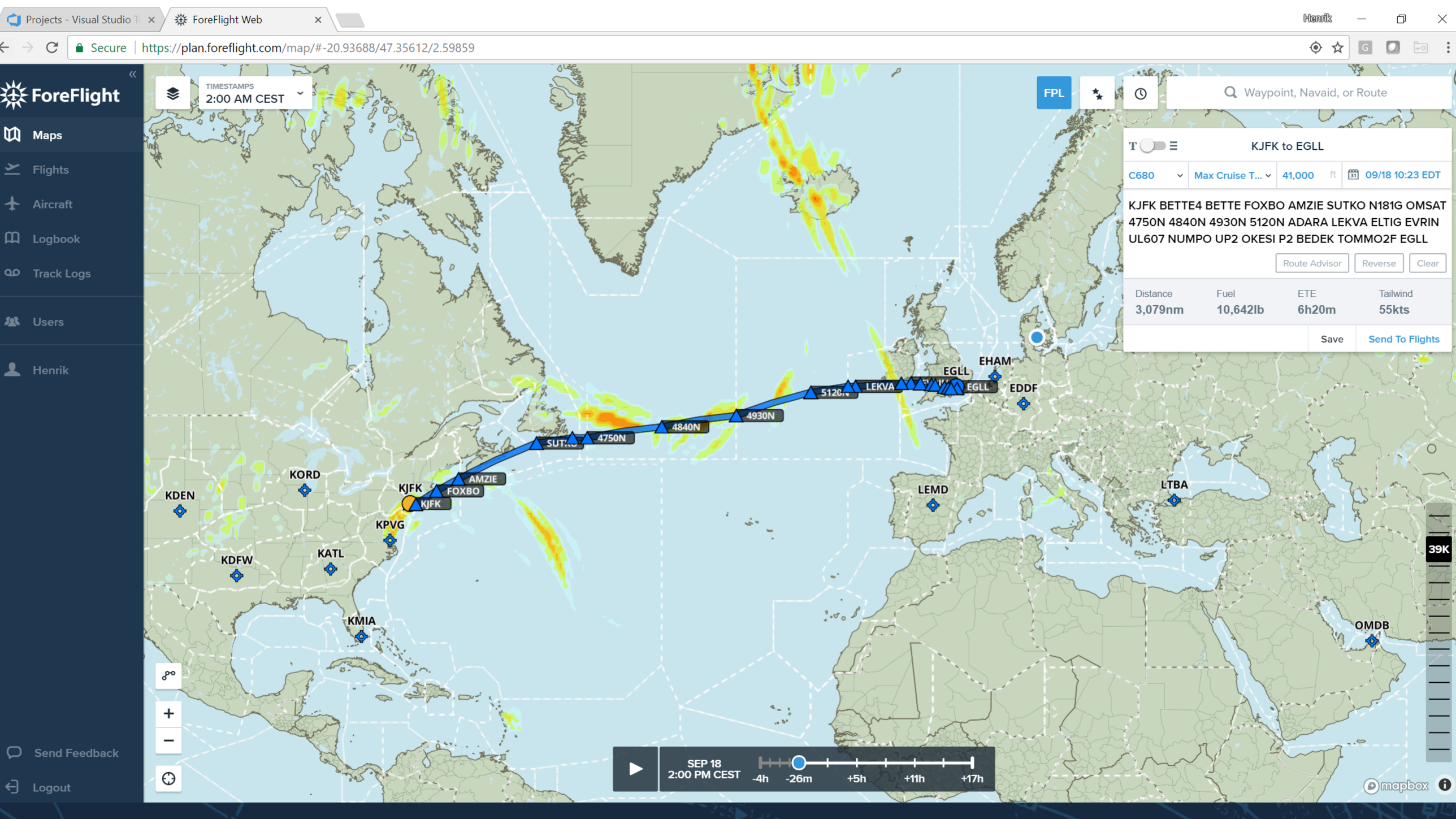
About me

- Education
  - Bogense skole, 1996-2006
  - Kold College HTX, 2006-2009
  - SDU, B.Eng, IT/Comm Engineer, 2009-2013
  - SDU, MS.c, Software Engineering 2013-2016
- Work
  - Student-dev/intern @ AIR SUPPORT 2011-2013
  - Lead-developer @ AIR SUPPORT 2013-2014
  - Head of Development @ AIR SUPPORT 2014-2015
  - CTO @ AviationCloud, 2012-Now
  - AviationCloud aquired by ForeFlight 2016

What does AC do?

- Build systems for *flight planning*
  - Route optimization
  - Fuel calculation algorithms
  - Air Traffic Control Communications systems
  - Runs in the cloud on 40+ servers
- 13 people in Denmark, 120+ globally

ForeFlight Coroporate Jet @ Houston, TX, USA


Projects - Visual Studio
ForeFlight Web
Henrik
Secure
https://plan.foreflight.com/map/#-20.93688/47.35612/2.59859
ForeFlight
Maps
Flights
Aircraft
Logbook
Track Logs
Users
Henrik
Send Feedback
Logout
TIMESTAMPS
2:00 AM CEST
FPL
Waypoint, Navaid, or Route
KJFK to EGLL
C680
Max Cruise T...
41,000
ft
09/18 10:23 EDT
KJFK BETTE4 BETTE FOXBO AMZIE SUTKO N181G OMSAT
4750N 4840N 4930N 5120N ADARA LEKVA ELTIG EVRIN
UL607 NUMPO UP2 OKESI P2 BEDEK TOMMO2F EGLL
Route Advisor
Reverse
Clear
Distance
3,079nm
Fuel
10,642lb
ETE
6h20m
Tailwind
55kts
Save
Send To Flights
KJFK
AMZIE
FOXBO
4750N
4840N
4930N
LEKVA
EGLL
EHAM
EDDF
KORD
KDEN
KPVG
KATL
KDFW
KMIA
LEMD
LTBA
OMDB
39K
SEP 18
2:00 PM CEST
-4h
-26m
+5h
+11h
+17h
mapbox

Whats an API?

- Application programmable interface.
- A well defined interface that allows other systems (or component within the same system) to connect
- Most often this means an HTTP endpoint where you can submit/get some data as JSON or XML.
- Relevant link: https://swagger.io/

Everything needs an API

- Internal APIs
  - Connect server with mobile apps
  - Connect one tier of servers to another
- External APIs
  - Reddit: API allows you to retrieve and create posts, retrieve user data etc
  - Google Maps API: Allow you to generate static maps, retrieve driving directions etc
  - Twitter API: Allows you to retrieve and post tweets
  - Essentially, most modern system has some kind of API

AviationClouds product: APIs for flight planning

- AviationCloud builds algorithms and systems for flight planning and sells them as APIs
- As we don't have much user interface work, a lot of our work is focused on non-functional requirements
  - High uptime
    Our system is a dependency and fundamental part of other systems, so when we go down, they go down.
  - High robustness
    We do not control the other systems – our system must be well tested and robust against different types of input
  - Generalization and adaption
    As we do not know the ultimate use-cases of our system and algorithms, things must be generalized enough to be able to adapt to a wide range of use-cases
  - ….

$ENV us-prod

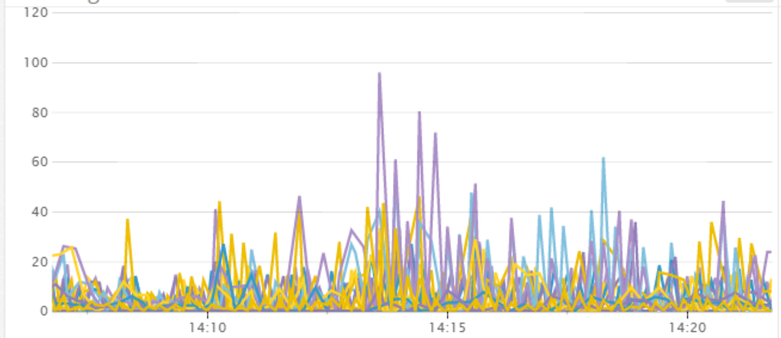
Austin 07:21:56 CDT

UTC 12:21:56 UTC

Europe 2:21:56 pm CEST

CPU Avg Host

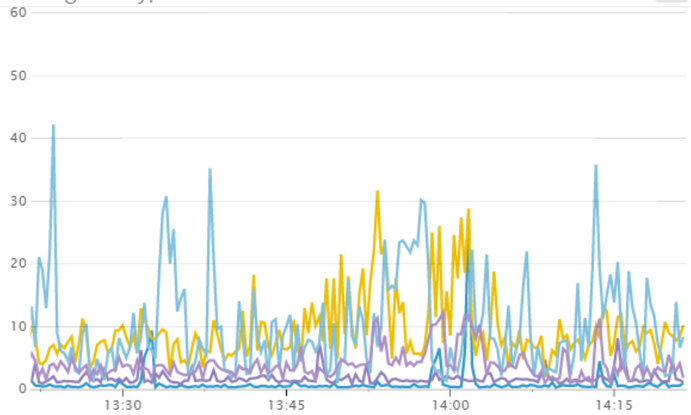
15m



0.19 %	Avg: 0.6…	system.cpu.user	{acenv:us-prod,host:usprod-bw0}
0.48 %	Avg: 0.6…	system.cpu.user	{acenv:us-prod,host:usprod-bw1}
1.42 %	Avg: 2.0…	system.cpu.user	{acenv:us-prod,host:usprod-cb0}
0.29 %	Avg: 6.6…	system.cpu.user	{acenv:us-prod,host:usprod-fw0}
0.29 %	Avg: 7.2 %	system.cpu.user	{acenv:us-prod,host:usprod-fw1}
13.16 %	Avg: 6.5…	system.cpu.user	{acenv:us-prod,host:usprod-fw2}
0.096 %	Avg: 6.8…	system.cpu.user	{acenv:us-prod,host:usprod-fw3}
4.25 %	Avg: 7.3…	system.cpu.user	{acenv:us-prod,host:usprod-fw4}

CPU avg RoleType

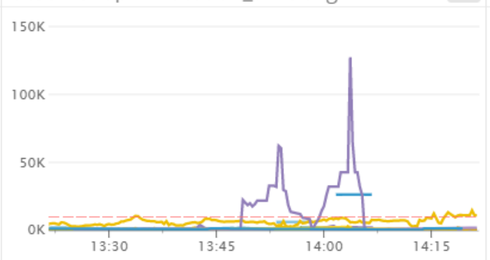
1h



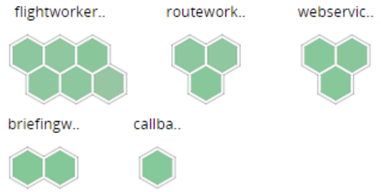
1 %	Avg: 0.9…	system.cpu.user	{acenv:us-prod,azure-role:briefingworkerrole}
1.14 %	Avg: 1.7…	system.cpu.user	{acenv:us-prod,azure-role:callbackworkerrole}
10.23 %	Avg: 9.3…	system.cpu.user	{acenv:us-prod,azure-role:flightworkerrole2}
8.34 %	Avg: 10.…	system.cpu.user	{acenv:us-prod,azure-role:routeworkerrole3}
1.84 %	Avg: 3.9…	system.cpu.user	{acenv:us-prod,azure-role:webservicerole}

acserver.api.invocation_time.avg

1h



system.cpu.user



Processed vs Yesterday

1h

15.03K	routecostcalculat…	-2.43K
6.89K	griddedweatherr…	-0.82K
4.01K	storedrouteretriv…	-1.6K
2.1K	v2flightcalculatio…	-0.21K
1.26K	extendedrouteco…	+0.16K
0.57K	briefinggeneratio…	-0.24K
0.45K	routegeneration…	+0.12K
0.04K	cfmuassistedrout…	No past value
0.011K	navretrievalmess…	No past value
0.004K	oceanictracksretr…	No past value
0.001K	bulkstoredrouter…	No past value
0.001K	weatherretrivalm…	No past value

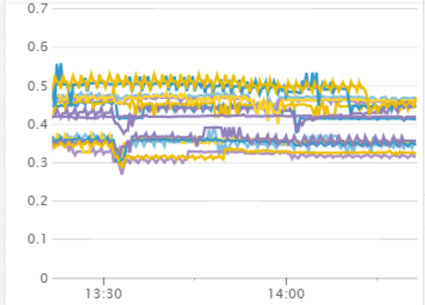
Response Time vs Yesterday

1h

9.84K	routegeneration…	+3.19K
0.54K	v2flightcalculatio…	+0.2K
1.97K	storedrouteretriv…	+0.15K
0.18K	briefinggeneratio…	+0.11K
0.38K	routecostcalculat…	+0.086K
0.19K	griddedweatherr…	-0.028K
6.95K	cfmuassistedrout…	-0.42K
2.02K	extendedrouteco…	-0.45K
1.3K	bulkstoredrouter…	No past value
0.59K	weatherretrivalm…	No past value
0.12K	navretrievalmess…	No past value
0.28K	oceanictracksretr…	No past value

system.mem.pct_usable

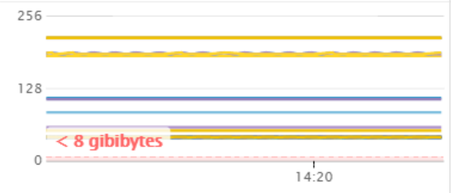
1h



0.41	Av…	system.me…	{acenv:us-prod,host:us…
0.42	Av…	system.me…	{acenv:us-prod,host:us…
0.45	Av…	system.me…	{acenv:us-prod,host:us…
0.34	Av…	system.me…	{acenv:us-prod,host:us…
0.32	Av…	system.me…	{acenv:us-prod,host:us…
0.35	Av…	system.me…	{acenv:us-prod,host:us…
0.35	Av…	system.me…	{acenv:us-prod,host:us…

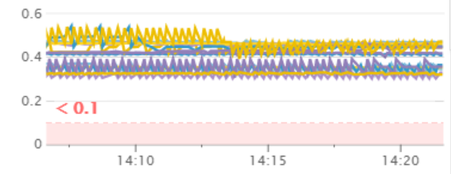
Alert: AviationCloud :: Free diskspac…

OK



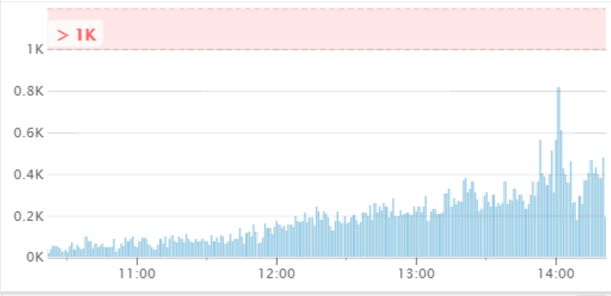
Alert: AviationCloud :: Low memory …

OK



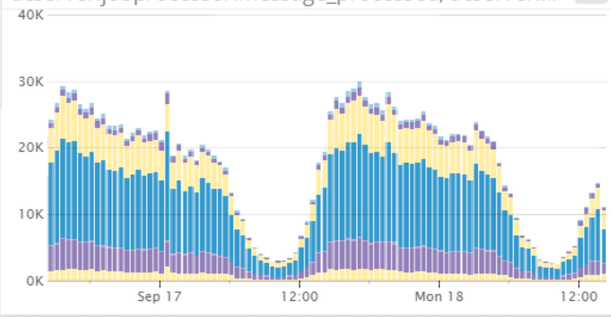
Alert: AviationCloud :: API :: Possible Overload.

OK



acserver.jobprocessor.message_processed, acserver….

2d



API requests Last 24h

1d

187.83K	retrievegriddedweather
133.03K	retrievestoredroutes
122.29K	calculateroutemetrics
30.57K	calculateflightplan
18.15K	generatebriefingpackage
11.65K	resolveroute
10.29K	generateroutes
0.99K	validateflight
0.76K	resolveaftnadresses
0.55K	fileflightplanasynchronously

system.uptime

1h

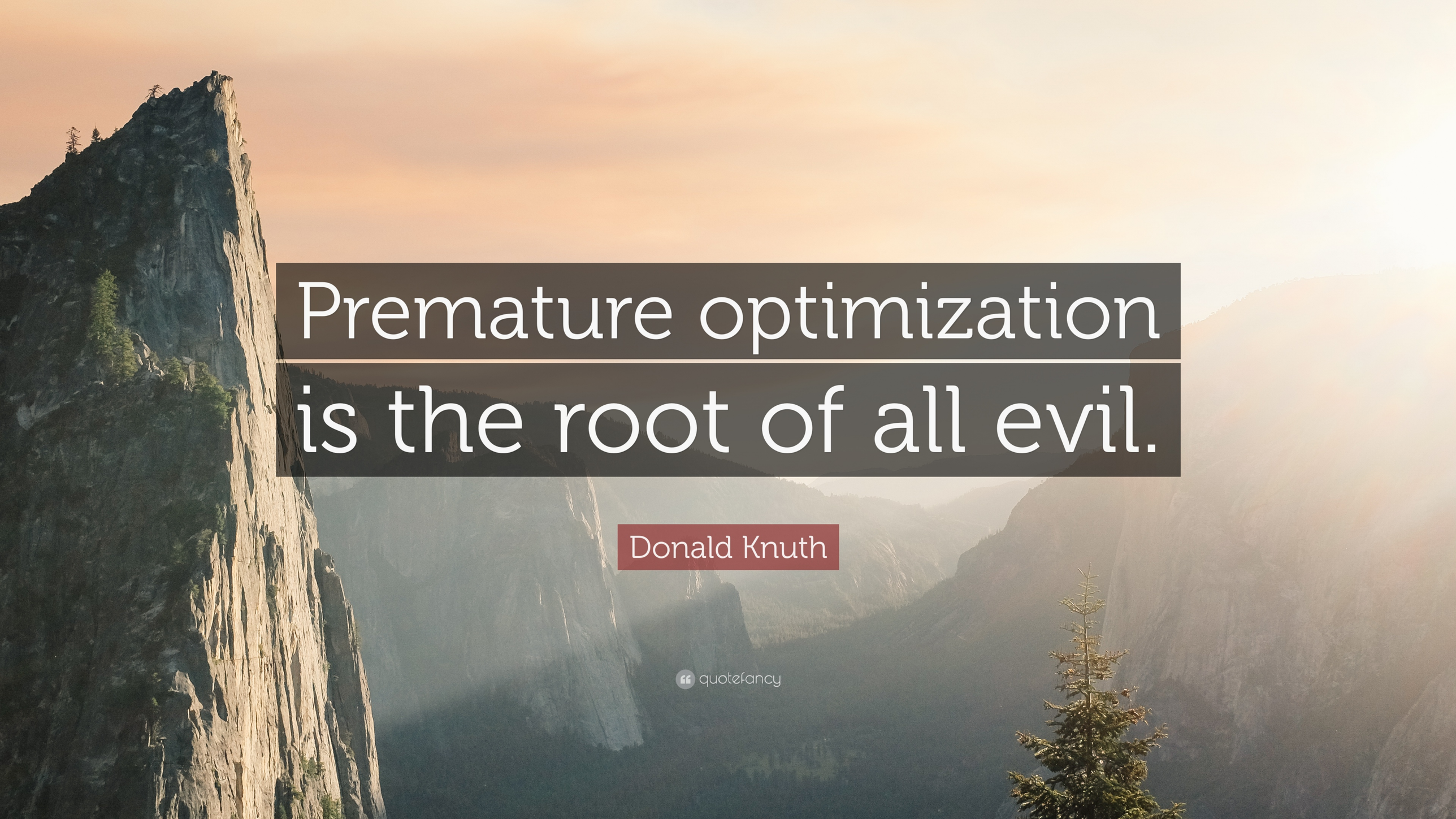
hr

1.18	usprod-fw5
27.29	usprod-fw0
36.28	usprod-cb0
36.28	usprod-bw0
36.28	usprod-bw1
36.28	usprod-web1
36.28	usprod-web0
36.28	usprod-fw3
36.28	usprod-fw1
36.29	usprod-fw4
36.29	usprod-fw2
36.29	usprod-rw2
36.29	usprod-rw0
71.39	usprod-web2
71.39	usprod-rw1

Software architecture is a mixed field

- Its easy to throw around a lot of buzzwords without having any idea of what you're talking about
- If neglected, it can lead to unmanageable technical debt, leading to unmaintainable and unreliable systems
- If overdone, it leads to inefficient development, ultimately making those systems fail to meet their functional requirements
- Therefore, two guiding principles … :)


Talk is cheap.
Show me the code.
Linus Torvalds
quotefancy


Premature optimization is the root of all evil.
Donald Knuth
quotefancy

Flight plan filing – An introduction

- Back in the 1960's the aviation industry was booming because of the invention of jet aircrafts
- Countries needed a way to communicate about flights – and this was before the internet
- The solution: Aeronautical Fixed Telecommunication Network, AFTN

The Boeing 707
Launched in 1958 marked the beginning of the jet-age

An AFTN message

- AFTN: Maximum, max 69 characters per line, max 1104 characters
- Below is an example of a flight plan message (FPL) describing a flight from Malage to Copenhagen
- These messages are distributed to every country along the route of flight and all relevant airports


```
ICAO FPL: (FPL-TAILNO-IG
          -SR22/L-GLORUVXWY/N
          -LEMG1402
          -N0174F170 BLN1L BLN/N0172F160 G5 CJN R10 PPN R299 BEGUY/N0174F170
          DCT SOVOS DCT BASTO DCT SECHE DCT BUGUS DCT TUGLI DCT LASEV DCT
          RISUN/N0172F160 G21 GIVOR V36 AKELU N852 DIK/N0174F170 N853 ARCKY
          Z907 ELDAR Z727 HMM M170 STADE L23 HAM P605 GESKA Z711 MONAK MONAK1D
          -EKCH0835
          -PBN/B2 DOF/170918 EET/LFBB0252 LFFF0432 LFEE0459 EDGG0555 EBBU0556
          EDGG0609 EBBU0610 EDGG0620 EBBU0620 EDGG0622 EDWW0701 EKDK0805)
```

Flight plan filing system

- Our system generates messages which gets sent to air traffic control, we denote these outgoing messages
- We (might) receive replies back to these messages conveying important information about the flight. We call these incoming messages
- Outgoing messages has an internationally standardized format
- Incoming messages messages do not

Some bad situations

- If the system is down → Pilots cannot submit flight plan → Aircraft cannot take off → $$$$$
- If the system fails to send an outgoing message → Aircraft might be rejected entrance into an airspace → $$$
- If the system looses an incoming message → Pilot might not get important information → $$$ or worse

Functional and non-functional requirements

- Functional requirements describes what the system should do from a user point of view
  - The system should allow the user to input X and save that information
- Non functional requirements describe attributes about the system that are not directly visible to the user
  - The system should be able to handle at least X requests per second
- Non-functional requirements are often some of the main drivers when designing software architechture

Functional requirements

- The user must be able to submit a flight plan (shown on right) to the system
- The user must be able to update an existing flight plan
- The user must be able to cancel a submitted flight plan
- The user must be able to view an existng flight plan
- The user must be able to view all messages related to a flight plan

INTERNATIONAL FLIGHT PLAN

PRIORITY
<<≡FF ⇒
ADDRESSEE(S)
KZNYZQZX EUCHZMFP EUCBZMFP CZQMZQZX
<<≡
FILING TIME
ORIGINATOR
<<≡
SPECIFIC IDENTIFICATION OF ADDRESSEE(S) AND (OR) ORIGINATOR

3 MESSAGE TYPE
<<≡ (FPL
7 AIRCRAFT IDENTIFICATION
— OYACC
8 FLIGHT RULES
— I
TYPE OF FLIGHT
G <<≡
9 NUMBER
—
TYPE OF AIRCRAFT
GLF5
WAKE TURBULENCE CAT
/ M
10 EQUIPMENT
— GLORUVXWY / N <<≡
13 DEPARTURE AERODROME
— KTEB
TIME
1400 <<≡
15 CRUISING SPEED
— N0490
LEVEL
F390
ROUTE
⇒ DCT SANTT J42 BOS DCT BAXIE DCT SCOTS N315A
ELSIR/M085F390 DCT 50N050W DCT 52N040W DCT 53N030W DCT 54N020W DCT
DOGAL/N0486F390 DCT BEXET DCT KORAK DCT DONEB DCT WST DCT OP DCT
LIFFY UL975 WAL LOREL4F
<<≡

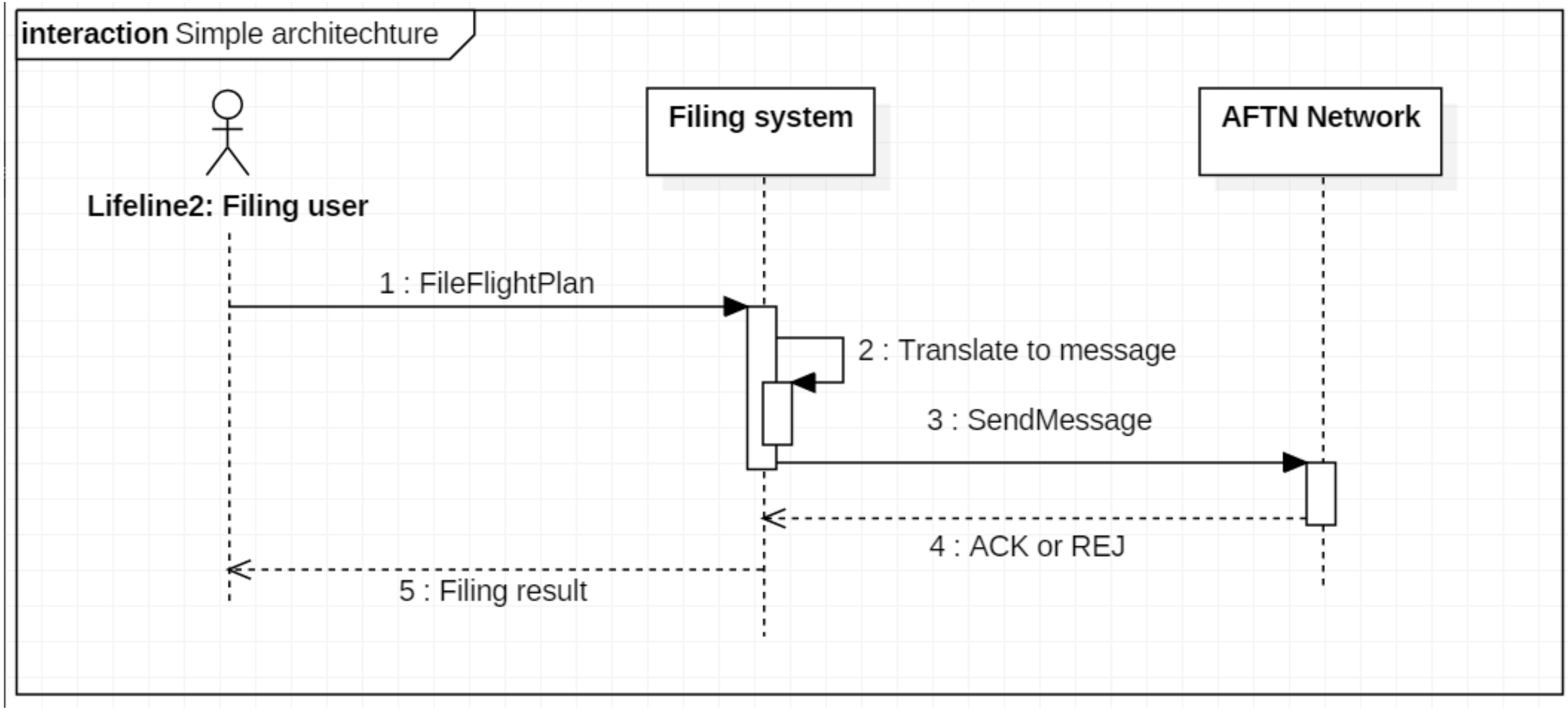
TOTAL EET
16 DESTINATION AERODROME
— EGGW
HR MIN
0543
ALTN AERODROME
⇒ EGCC
2ND ALTN AERODROME
⇒ <<≡
18 OTHER INFORMATION
— PBN/B2 DOF/170809 EET/KZBW0002 CZQM0042 CZQX0123 EGGX0322 EISN0424
EGTT0505
) <<≡

SUPPLEMENTARY INFORMATION (NOT TO BE TRANSMITTED IN FPL MESSAGES)
19 ENDURANCE
HR MIN
— E / 0648
PERSONS ON BOARD
⇒ P / TBN
EMERGENCY RADIO
UHF VHF ELT
⇒ R / X X X
SURVIVAL EQUIPMENT
POLAR DESERT MARITIME JUNGLE
⇒ X / X X M X
JACKETS LIGHT FLUORES UHF VHF
⇒ X / X X X X
DINGHIES
NUMBER CAPACITY COVER COLOUR
⇒ X / ⇒ X ⇒ <<≡
AIRCRAFT COLOUR AND MARKINGS
A /
REMARKS
⇒ N / <<≡
PILOT-IN-COMMAND
C /) <<≡
FILED BY
ADDITIONAL REQUIREMENTS

Non-functional requirements

- #NF1: The system must support an uptime of at least 99.9 %
- #NF2: The system must ensure message integrity – that is, outgoing and incoming messages may never be lost
- #NF3: The system must continue to be able to accept flight plans even when AFTN network connectivity is unavailable
- #NF4: There can only be one single connection to AFTN, and that connectin must be routed through a single IP address on a server sitting on a protected network
- #NF5: Only one message can be sent to AFTN at a time

Design #1

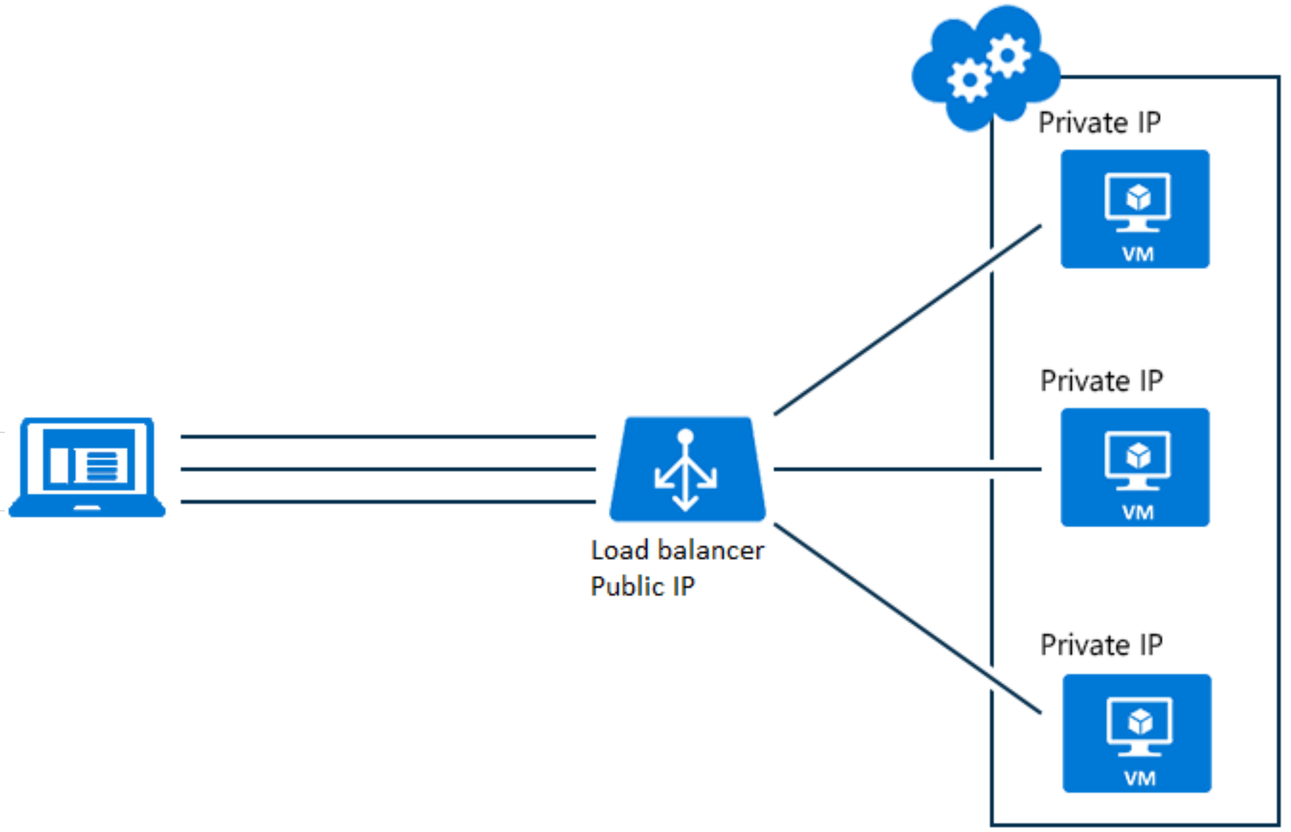

interaction Simple architechture
Lifeline2: Filing user
Filing system
AFTN Network
1 : FileFlightPlan
2 : Translate to message
3 : SendMessage
4 : ACK or REJ
5 : Filing result

Consequences of this design

- #NF5 dictates we can only send one message at a time to AFTN, so with this design, we can only allow one user to file a flight plan at a time
  - Result: Low throughput
- #NF4 dictates we can only make one connection from one single server with a dedicated IP address sitting on a protected network
  - If the network is protected, how can the user connect?
- #NF1 requires 99.9 % uptime. If user connect to a single server, how can we ensure that?
  - Software updates, OS updates, physical hardware maintainance etc…

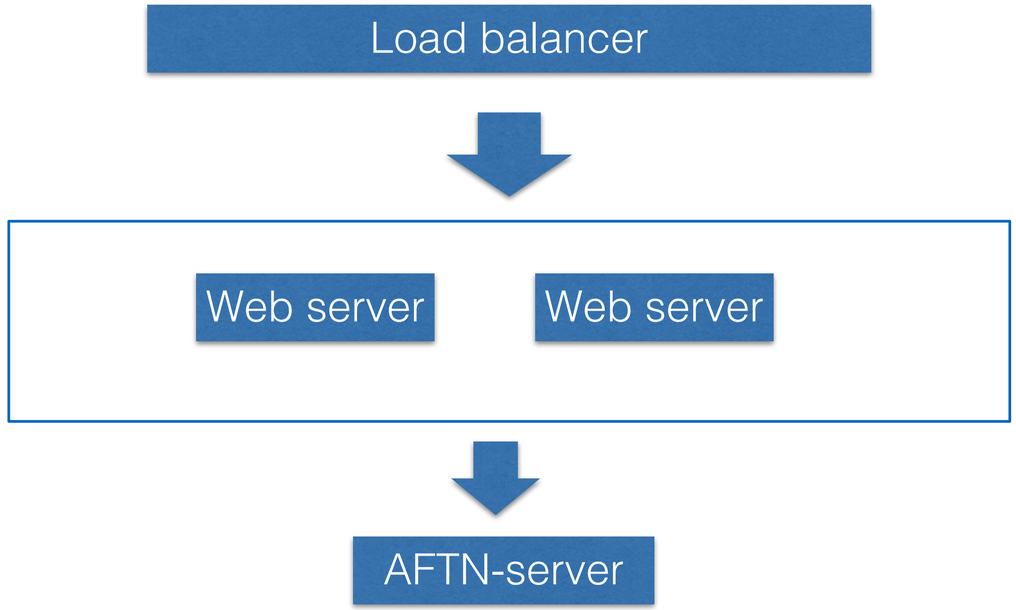
Solving availability – Load balancers

- A load balancer distributes load between servers
- They also help ensure uptime. One server can be disconnected from the load balancer during maintainance, connected once done, then the other is updated – aka ”Rolling updates”



Design #2 – Multi-tier

- Users (systems) connect to the load balancer
- Load balancer forward the request to a web server
- The web server sends the AFTN message by invoking an API on AFTN-server

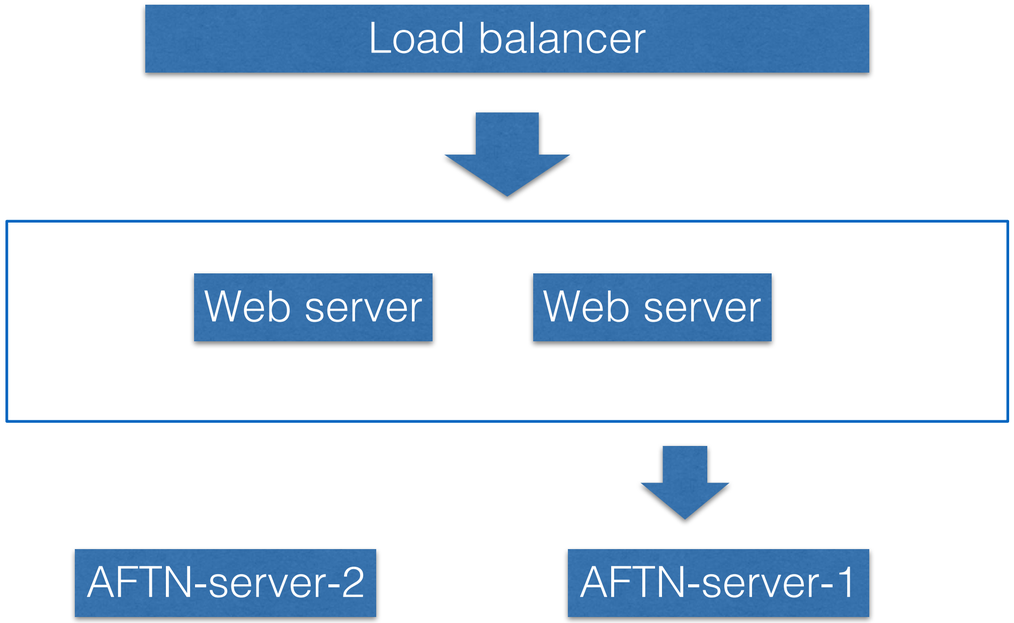


Design #2 - Issues

- Still limited throughput
- Difficult to implement the sent-one-message-at-a-time #NF5
- We solved the availability issue at the web-layer, but what if AFTN-server needs maintenance?
  - We can’t do load balancing, since only one active AFTN connection is allowed

Design #3 – Multi-tier + hot-standby AFTN

- A hot standby is a server that is inactive, but is ready to take over at any moment when it discovers the other server(s) is down
- In practice:
  - Active server registers a ”ping”
  - Non-active server looks for the ping
  - When non-active server sees missing ping, it makes itself active
  - Issue: Race conditions, but that’s a story for later ;-)



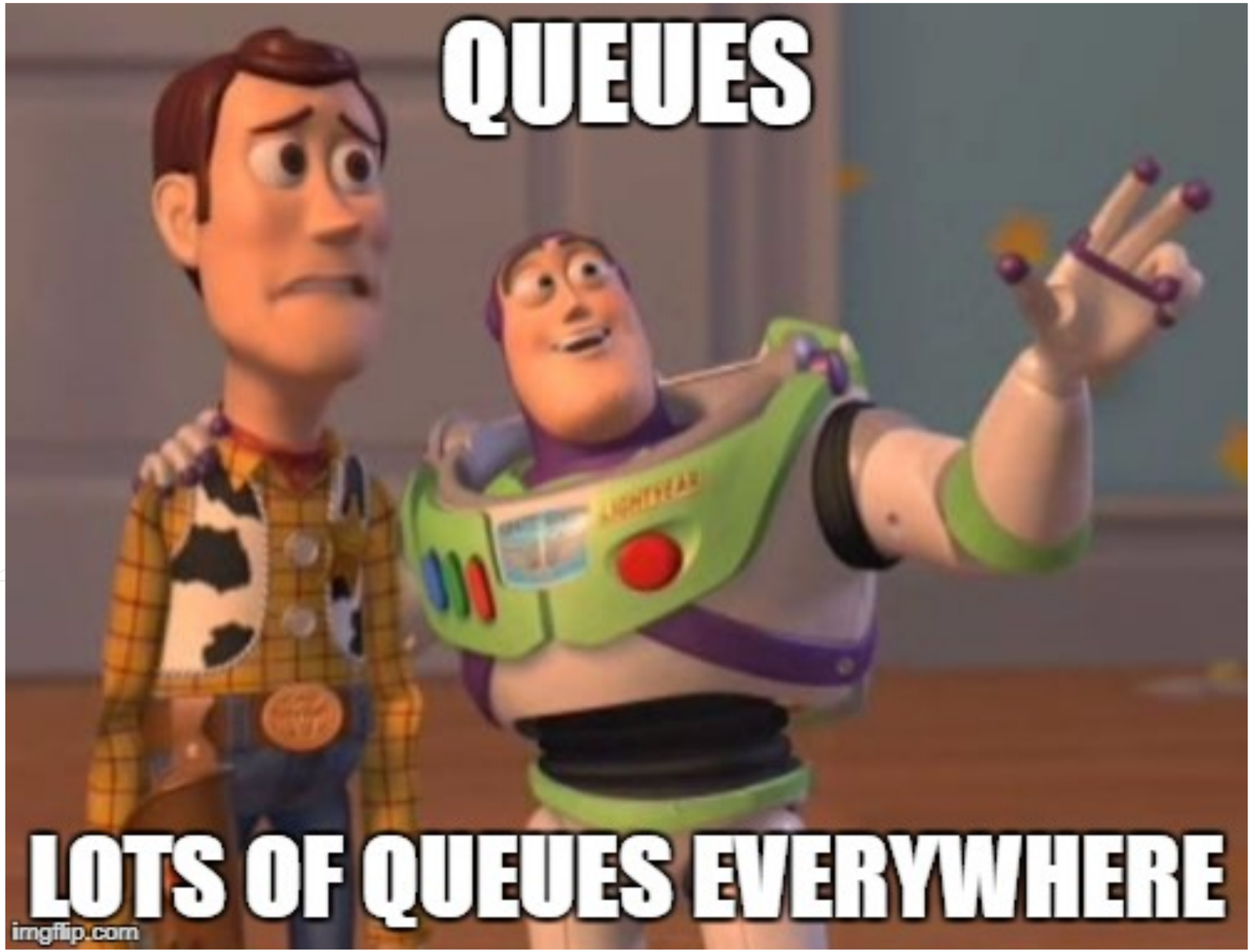
If AFTN-server 1 fails, web servers sends requests to AFTN-2 and vice-verca

Issues – Design #3

- Things are getting better on the availability side. Using load-balancing and hot-standby we should be able to ensure that our own services keeps running
- We still have some major issues:
  - How can we continue to accept flight plans if AFTN is unavailable?
  - How do we ensure message integrity?

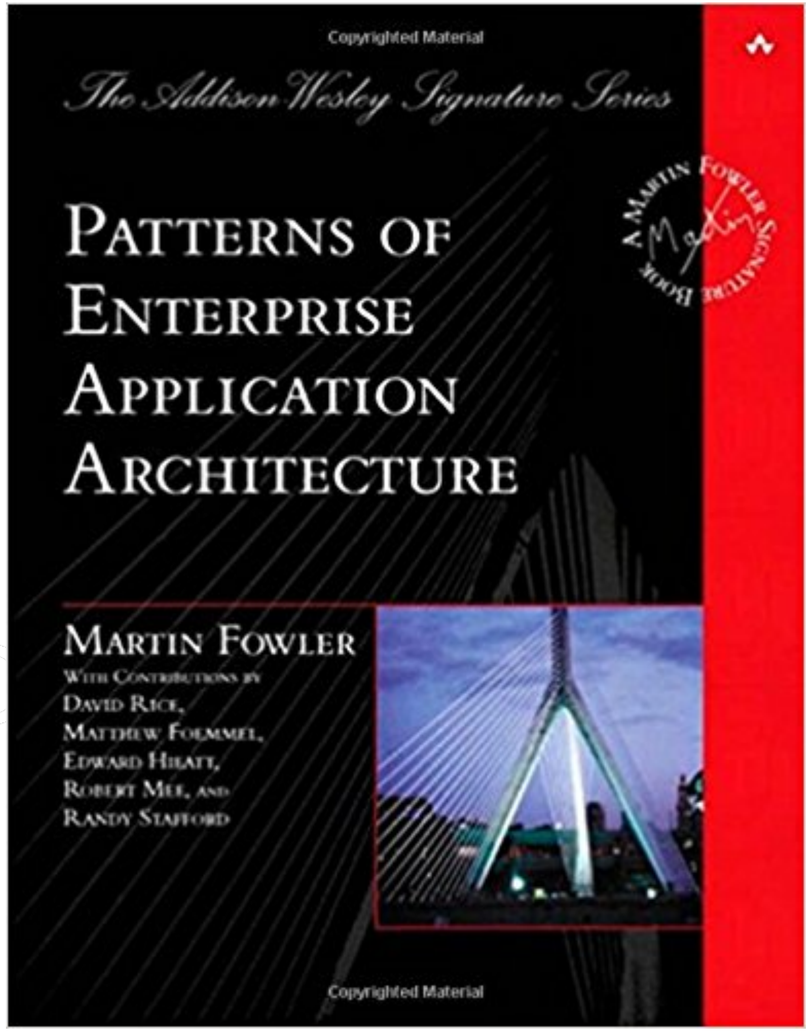
Queues

- Queue is a basic data structure that everyone should know
- They come as in-memory variant (i.e. Queue<T> in Java/C#)
- And as *distributed* and *persistent* queues


QUEUES
LOTS OF QUEUES EVERYWHERE
imgflip.com

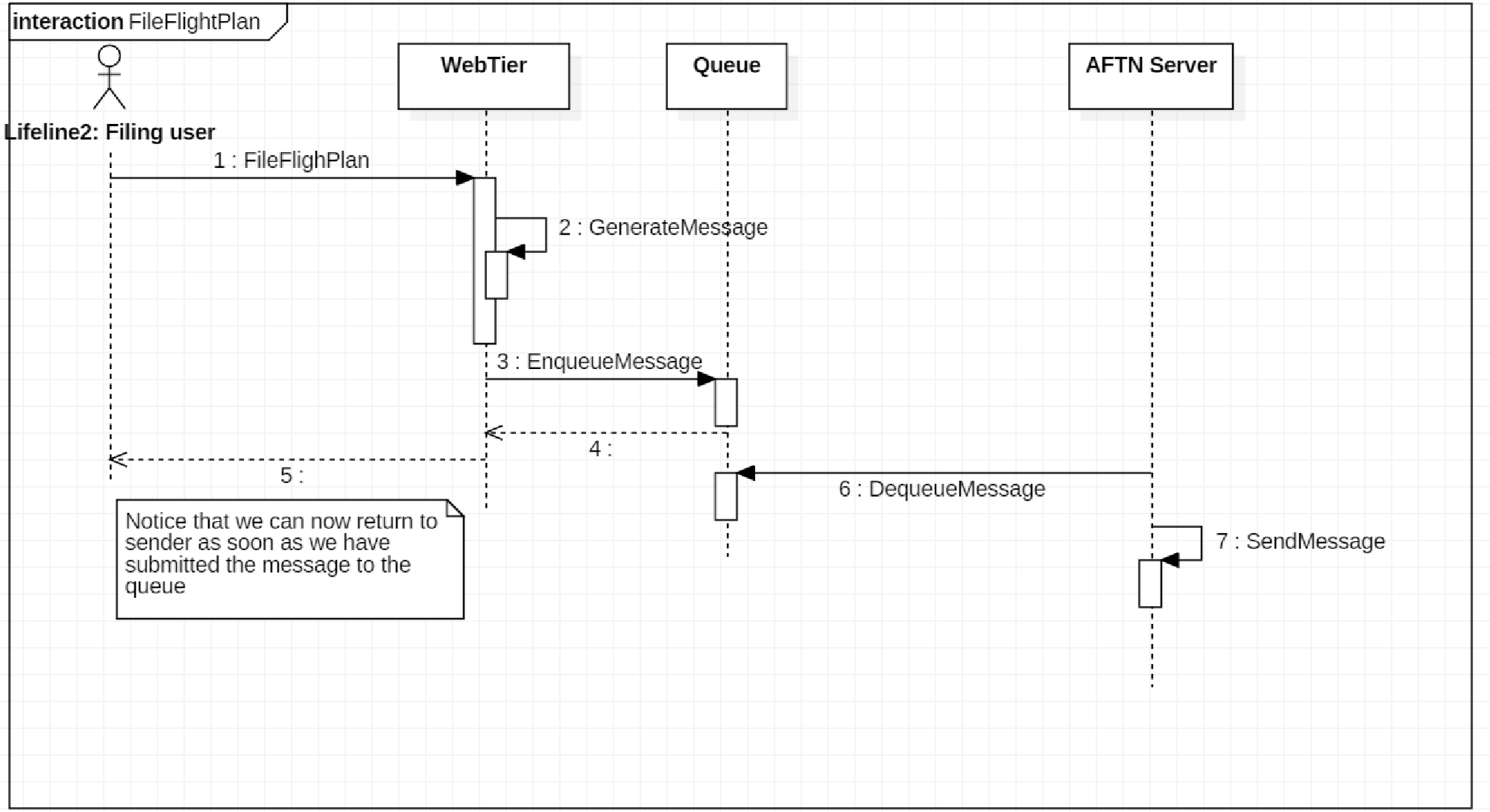
Distributed queues

- A distributed queue is a service running on a server with its only purpose of hosting a queue
- Other servers can enqueue or dequeue messages
- Distributed queues are often persistant, meaning that even if the server hosting the queue service becomes unavailable, the state will be restored once restarted



Explains some of the patterns around using queues as an architechtural building block

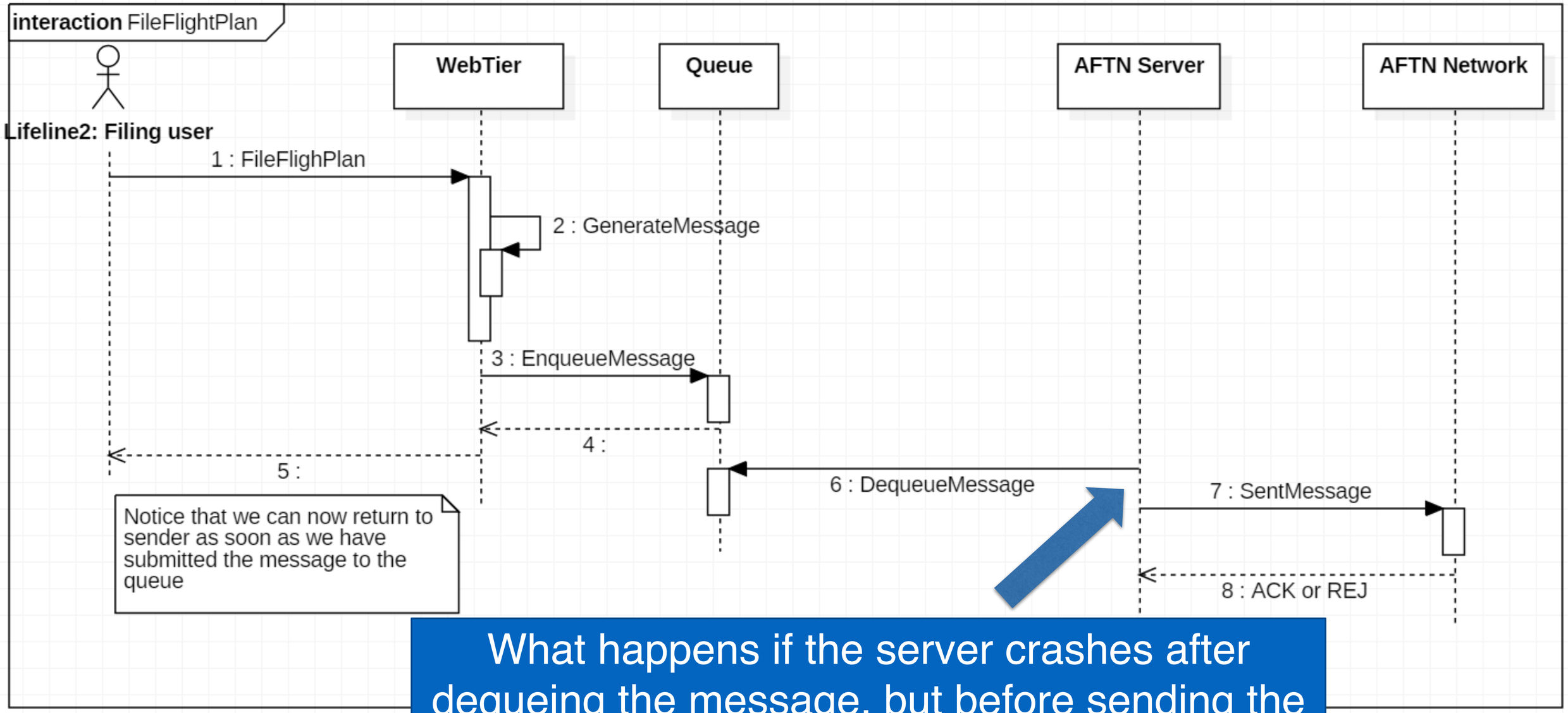
Design #4 – Queue based



Design #4

- We just made some big advances:
  - Messages can now be delivered to the system even when AFTN connectivity is down
  - Coordinating that only a single message can be sent is now easy because of the queue structure (first in, first out!)
  - If servers crashes, the messages are kept in the queue… Or are they?’
  - The only external resource the AFTN server needs to be able to access is the queue service, meeting the security requirements

Issues with design #4



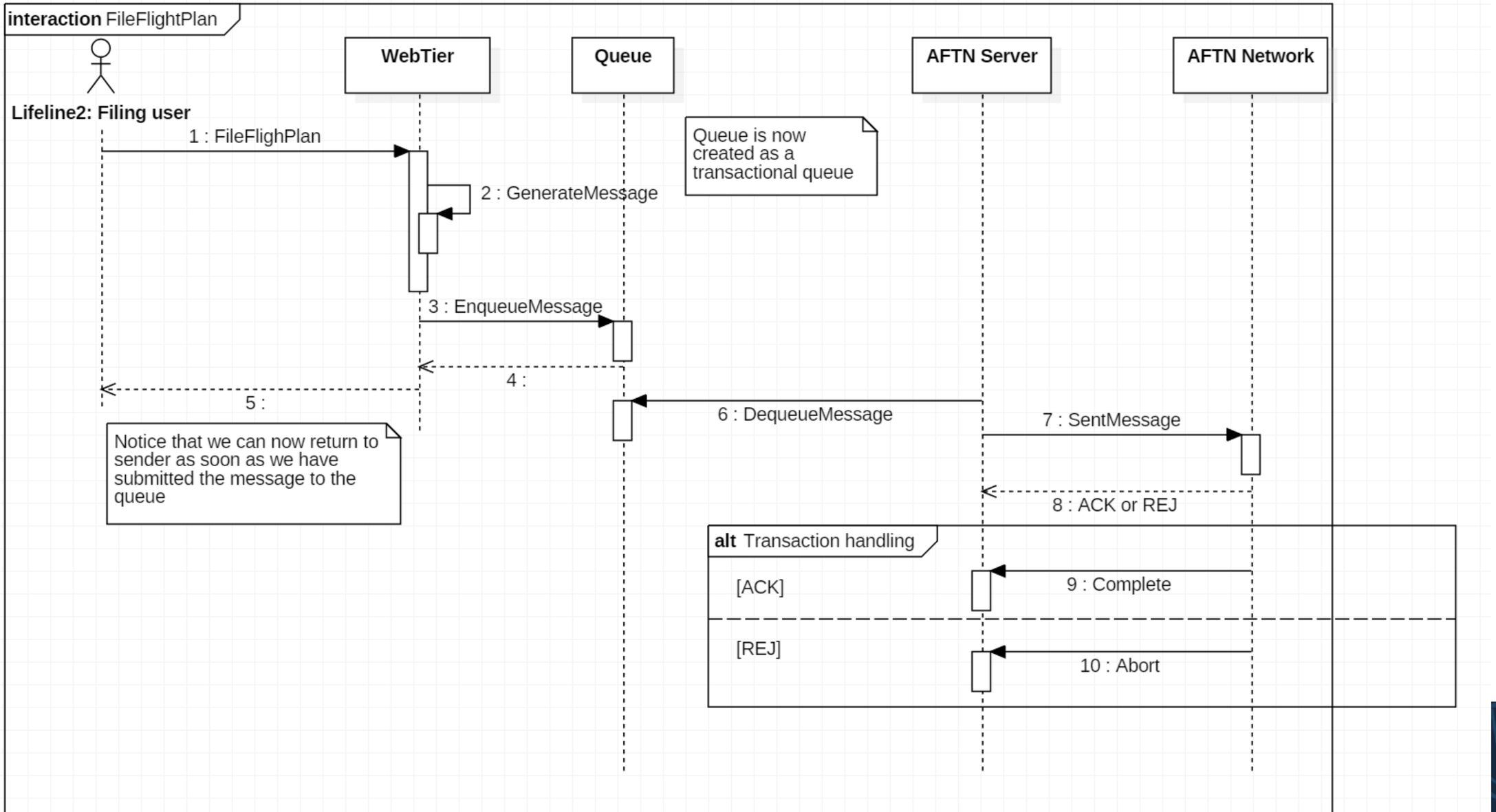
What happens if the server crashes after dequeing the message, but before sending the message?


EVERYTIME YOU LOOSE AN AFTN MESSSAGE
A KITTEN DIES
imgflip.com

Transactional systems

- A *transactional* system is a system where an operation can be rolled back if it does not succeeed
- In practice
  - You begin an operation by performing an operation such as ”Dequeue”.
  - You then perform the action.
  - If it suceeds, you end the transaction by calling a ”Complete” operation
  - If it fails, you abort the transaction by calling an ”Abort” operation
  - If the server does not recieve either a Complete or Abort operation, it assumes the operation has failed

Design #5 – Transactional queues



So what did we do?

- We introduced **load balancers** to help with availability
- We introduced **multi-tier** architecture to help with security and availability
- We introduced **hot-standby** for availability given the security requirement
- We introduced **queue** to help with availability and increase decoupling to the AFTN network itself
- We introduced the concept of **transactions** to help with integrity

- All of this is to cope with requirements that are mainly non-functional!

A few final thoughts

- This just showed some of the design choices made to comply with non-functional requirements in the AviationCloud filing system
- Architecture is a balance between foreseeing what is required and not ending up making overkill solutions
- ”Its future Henriks problem” aka agile development :-)
- On the other hand, architechtural changes are some of th emost challenging ones for a production-system.