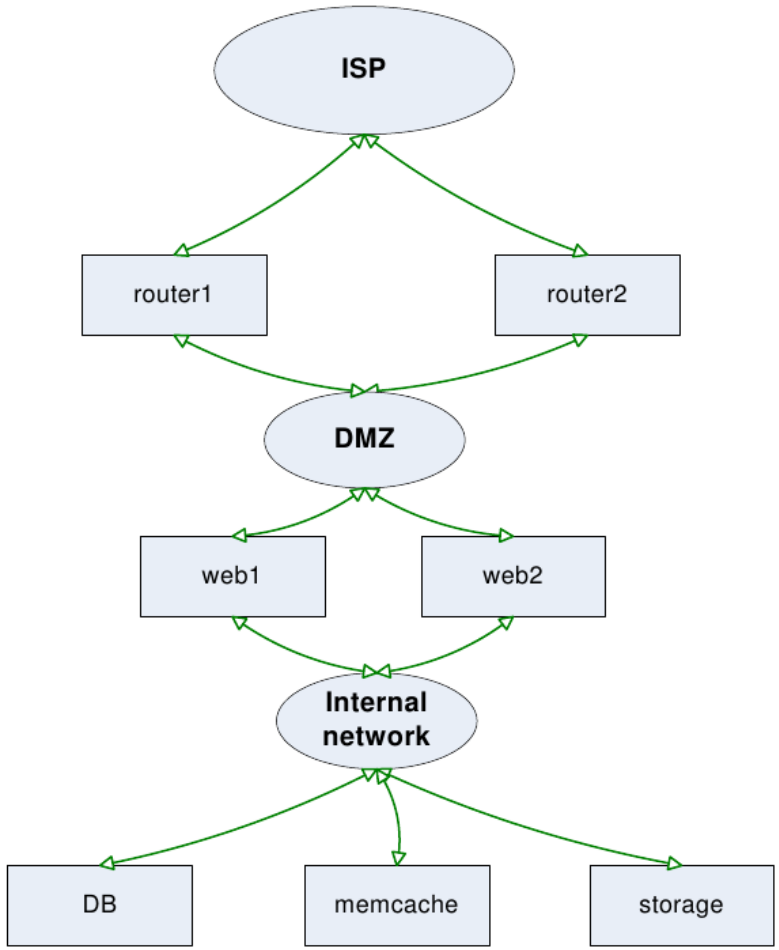
Избягване на single points of failure

или по-точно “Софтуерни и административни методи за избягване на хардуерни SPOF”

Васил Колев vasil@ludost.net

За какво става въпрос



Авиационен модел

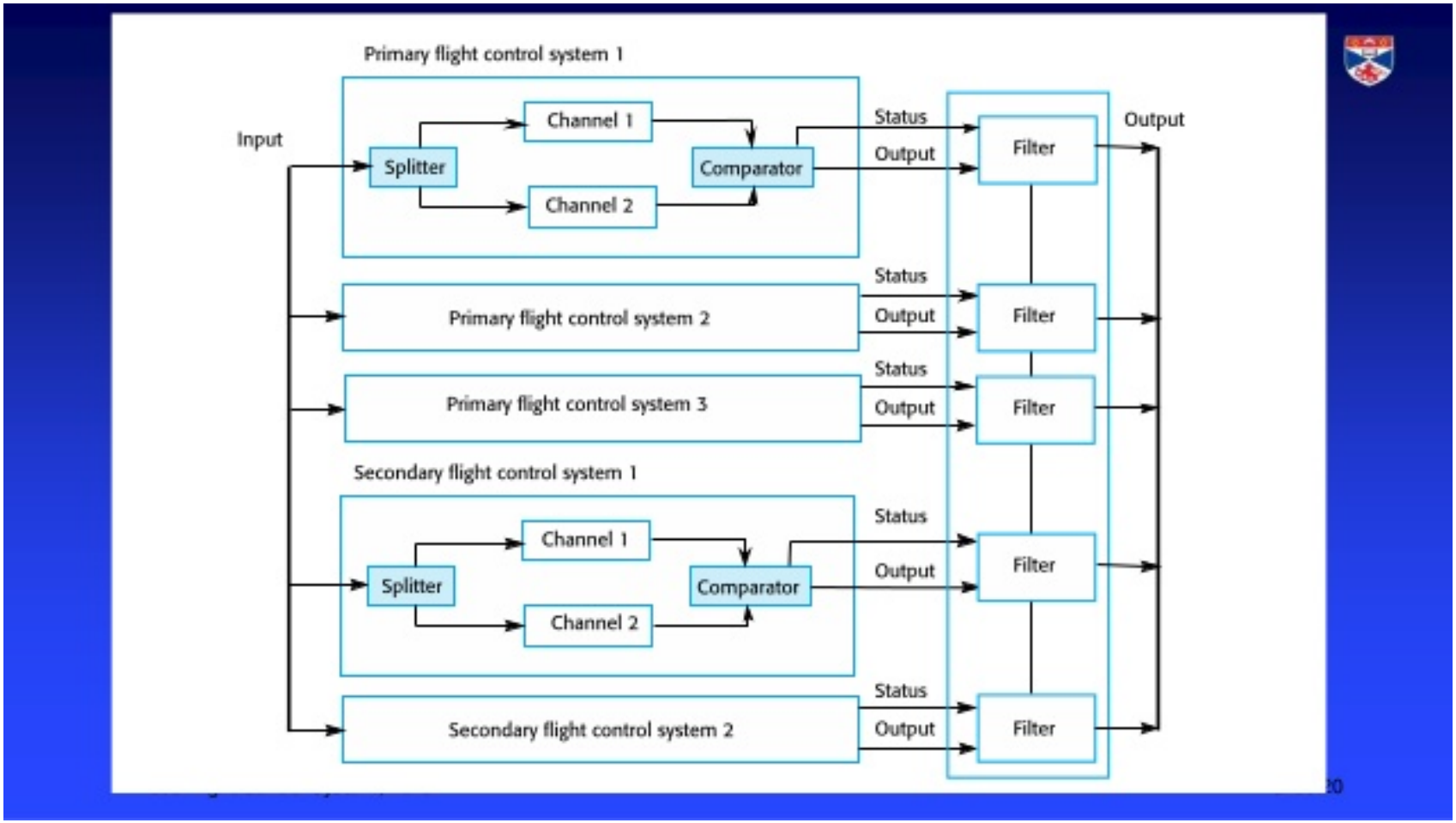


Рис.: aviation

Авиационен модел

- Адски скъп
- Ако системата не работи, умират хора
- Доста сложен
  - Airbus A380 има 530км кабели в себе си

Три неща, за които няма да говоря

- Софтуерни
- Meatware
- Цена

Софтуерни

- Лавинообразни проблеми
- Никой не може да си позволи авиацононния модел
- Правим се, че ги няма

Meatware

- Bus factor

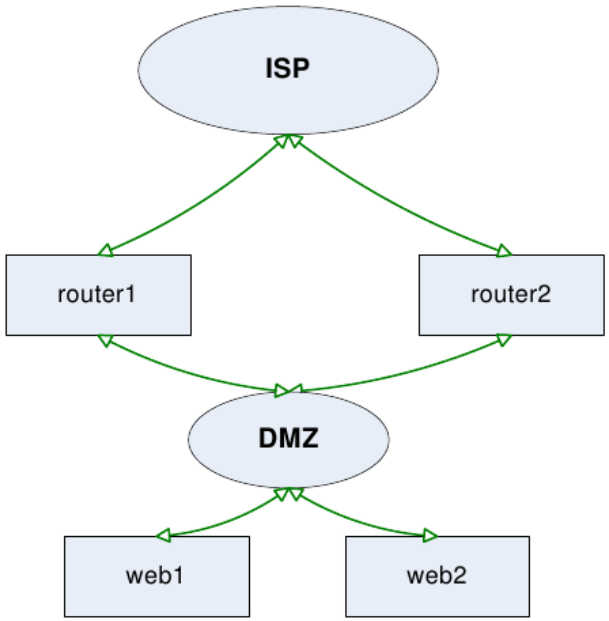
Цена

- Ваша работа си е кога си струва да премахвате SPOF-ове

State (състояние)

- Всичко, от машината на Тюринг насам има state
- Всички проблеми при избягване на SPOF-овете идват от него
- Няма такова нещо като stateless система

Примерна не-stateless система



Решения

- Do nothing
- Backup plan
- Cold/Hot spare
- Distribution/duplication

Do nothing

Backup plan

- Завиваме се в бял чаршаф и пълзим към гробищата

Cold spare

- По-добрият backup plan
- Все едно си държим резервни дискове или крушки на удобно място

Hot spare

- Работи синхронизирано с живата система
- Може да поеме работата ѝ
- Fencing
- Често срещано при базите данни

CAP теорема

- “Life sucks, and then you die”
- Consistency, Availability, Partition Tolerance
  - изберете си две
- Всички правят C+A системи в някакъв вид

Дистрибутиране

- Две или повече системи работят заедно
- . . . ако успеем да ги накараме
- Следват малко бележки по въпроса, теорията е огромна
  - и е silverbuletless

Split brain

- N сървъра
- Две части от ~ N/2 сървъра, които не се чуват помежду си
  - ... и си мислят, че са активни
- Известно още като “а имате ли как да си merge-нете омазаните данни”

“Арбитър”

- Повечето системи имат нужда от някой, който да поддържа state на живите компоненти
- Вариант 1 - система с hot spare
  - пример на следващия слайд
- Вариант 2 - система с дистрибутиран state и кворум
  - пример storpool :)

Load ballancers + web

- 2+ load ballancer-a, които си синхронизират state
  - или hot spare без синхронизация
- N web сървъра отзад, които обработват заявки и нямат собствени данни
- RAID масивите са еквивалентни на това като система

Hack-ът за eventual consistency

- Синхронната репликация е бавна, времето за един write >= латентността на най-бавната система
- Евентуална консистентност - update е валиден, ако са го приели X компонента
  - Най-често X = N/2 + 1
- Има incosinstent read от един компонент, и consistent от N-X+1 компонента

Благодаря за вниманието, въпроси?

- Благодаря за вниманието
- Въпроси?