Ingineria Calculatoarelor
Fault Tolerant Computing

- Cursul 0 -

Facultatea de Automatică și Calculatoare
Universitatea Politehnica București

“If anything can go wrong, it will.”

Murphy’s Law

About me

Dan Ștefan Tudose

- dan.tudose@upb.ro
- Office: ED422
- Office Hours: (almost) anytime on Teams
- https://ocw.cs.pub.ro/courses/iothings/dan.tudose
- Research & teaching:
  - Computer architecture, hardware/software interaction
  - Embedded and Pervasive Computing
  - Wireless Sensor Networks
  - Low Power Computing Architectures, Energy Harvesting
  - Fault tolerance
- Start-ups, Fitbit, Google

Important Stuff

Echipa laborator:

- Giorgiana Vlăsceanu
- Liviu Mitruță
- Theodor Ungureanu
- Dragoș Săndulescu
- Andreea Paiu

https://ocw.cs.pub.ro/courses/icalc

Notare:

- 1 punct - laborator (9 laboratoare)
- 2 puncte - proiect individual
- 2 puncte - lucrare de laborator (în săptămâna 14)
- 5 puncte - examen final

Cerințe pentru a promova:

- minim 6 prezențe la laborator
- minim 2.5 puncte din cele 5 puncte pentru activitatea laborator
- minim 2.5 puncte din cele 5 puncte din examenul final

The Curse of Complexity

Computer engineering is the art and science of translating user requirements we do not fully understand; into hardware and software we cannot precisely analyze; to operate in environments we cannot accurately predict; all in such a way that the society at large is given no reason to suspect the extent of our ignorance.[1]

Microsoft Windows NT (1992): ≈4M lines of code
Microsoft Windows 10 (2015): ≈70M lines of code

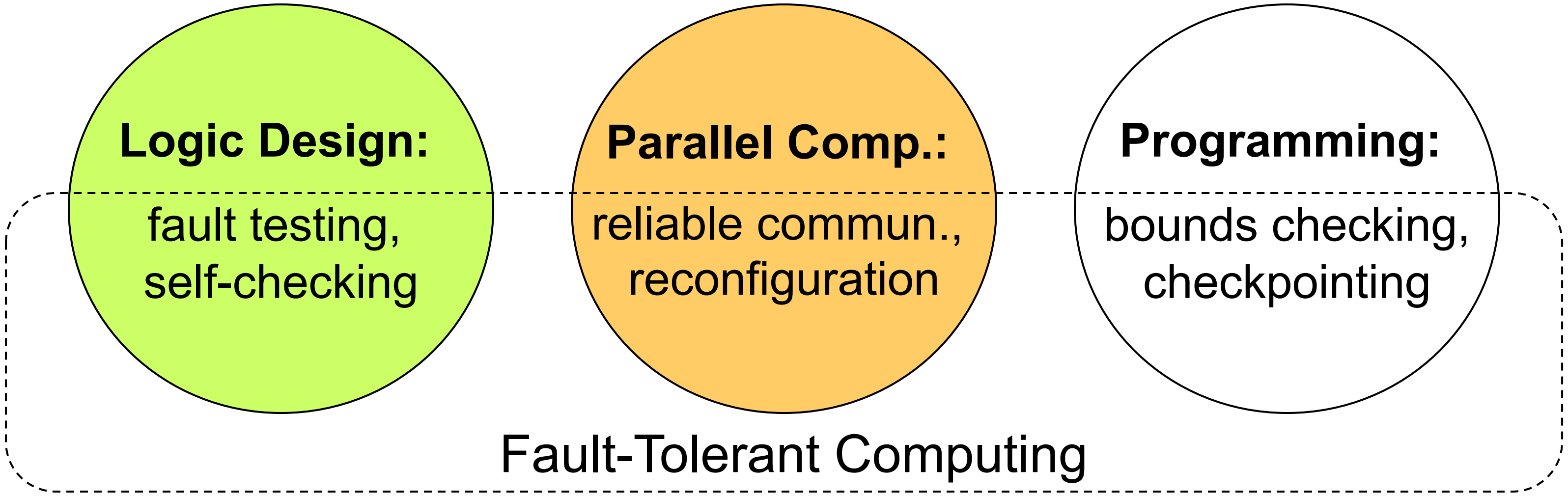
Intel Pentium processor (1993): ≈4M transistors
Intel Pentium 4 processor (2001): ≈40M transistors
Intel Xeon Haswell-EP (2015): ≈5500M transistors

[1]Adapted from definition of structural engineering: Ralph Kaplan, *By Design: Why There Are No Locks on the Bathroom Doors in the Hotel Louis XIV and Other Object Lessons*, Fairchild Books, 2004, p. 229

Why This Course Shouldn't Be Needed

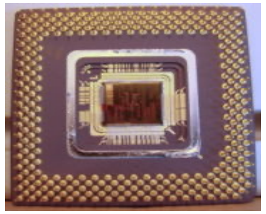
In an ideal world, methods for dealing with faults, errors, and other impairments in hardware and software would be covered within every computer engineering course that has a design component

Analogy: We do not teach structural engineers about building bridges in one course and about bridge safety and structural integrity during high winds or earthquakes in another (optional) course



Brief History of Dependable Computing

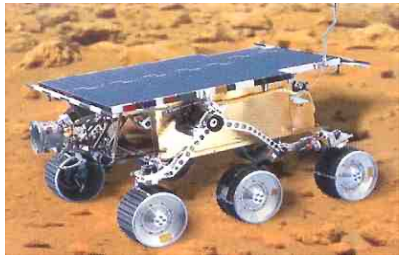
1940s:	ENIAC, with 17.5K vacuum tubes and 1000s of other electrical elements, failed once every 2 days (avg. down time = minutes)
1950s:	Early ideas by von Neumann (multichannel, with voting) and Moore-Shannon (“crummy” relays)
1960s:	NASA and military agencies supported research for long-life space missions and battlefield computing
1970s:	The field developed quickly (international conference, many research projects and groups, experimental systems)
1980s:	The field matured (textbooks, theoretical developments, use of ECCs in solid-state memories, RAID concept), but also suffered some loss of focus and interest because of the extreme reliability of integrated circuits
1990s:	Increased complexity at chip and system levels made verification, testing, and testability prime study topics
2000s:	Resurgence of interest owing to less reliable fabrication at ultrahigh densities and “crummy” nanoelectronic components



Dependable Computing in the 2020s

There are still ambitious projects; space and elsewhere

Harsh environments (vibration, pressure, temperatures)
External influences (radiation, micrometeoroids)
Need for autonomy (commun. delays, unmanned probes)

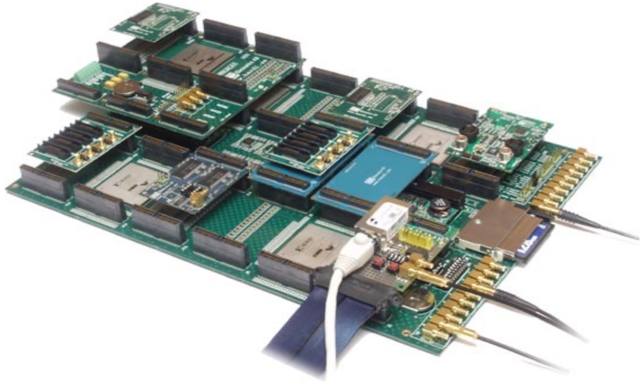


The need is expanding

More complex systems (e.g., system-on-chip)
Critical applications (medicine, transportation, finance)
Expanding pool of unsophisticated users
Continued rise in maintenance costs
Digital-only data (needs more rigorous backup)

The emphasis is shifting

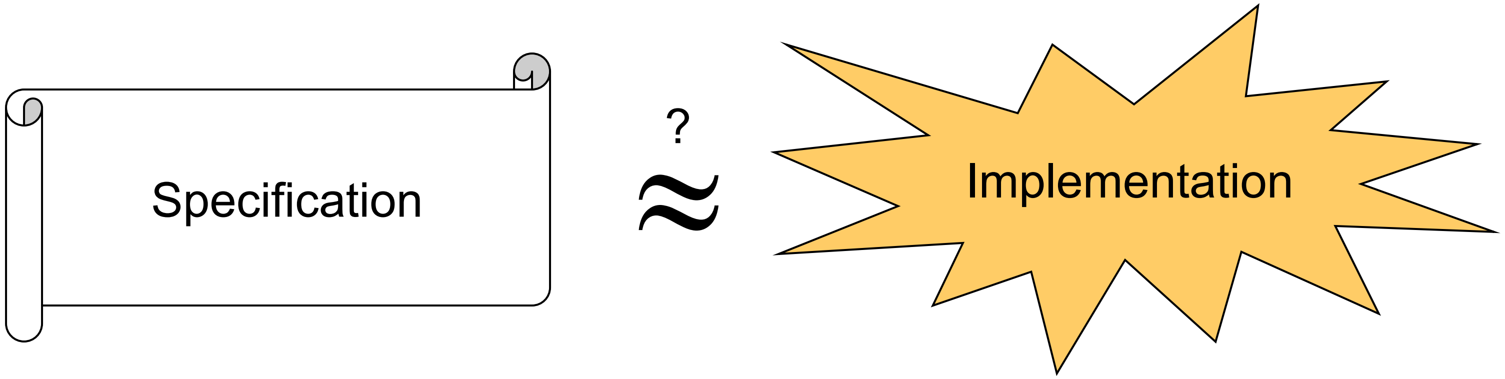
Mostly COTS-based solutions
Integrated hardware/software systems
Entire units replaced (little diagnosis)



Defining Failure

Failure is an unacceptable difference between expected and observed performance.[1]

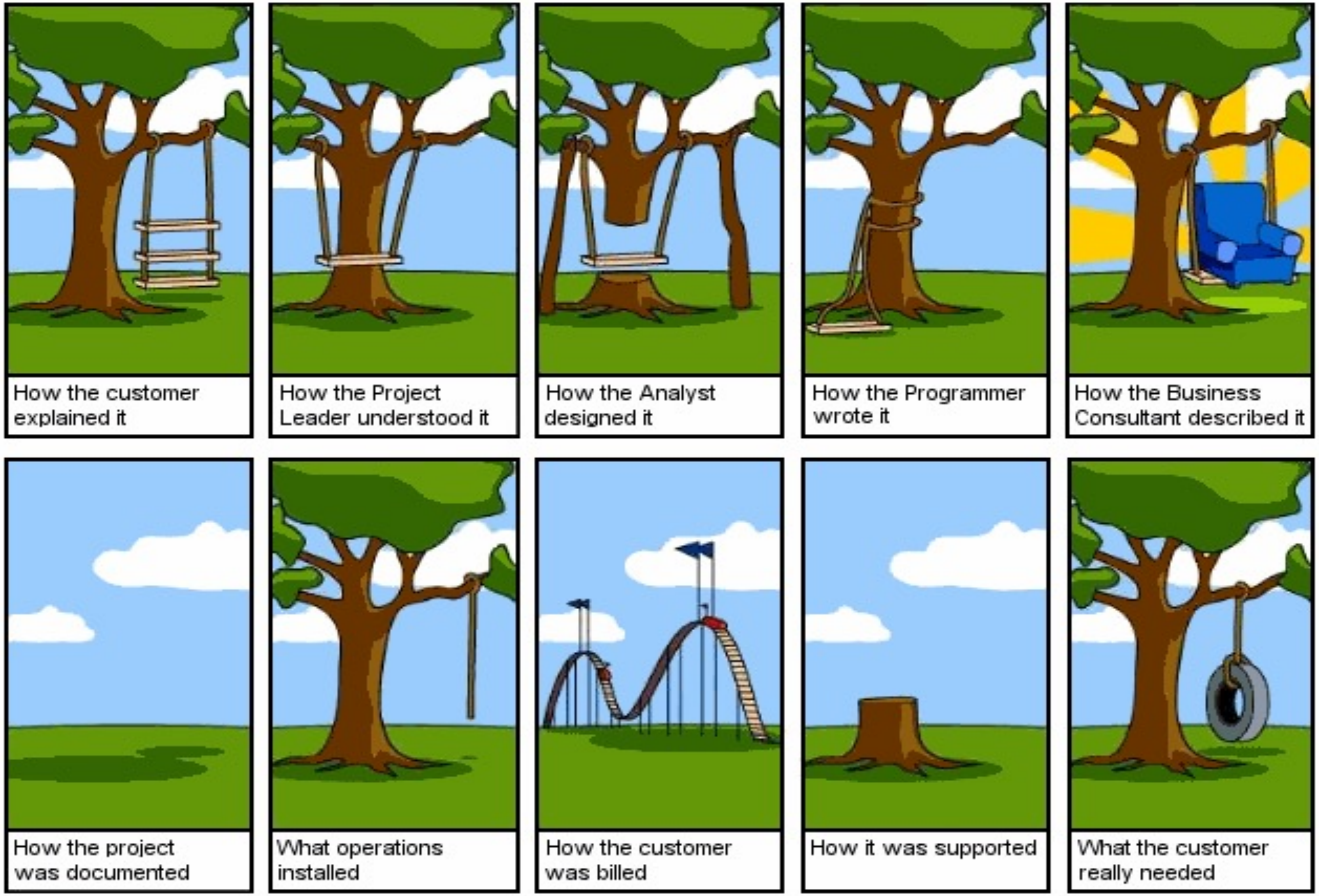
A structure (building or bridge) need not collapse catastrophically to be deemed a failure



Reasons of typical Web site failures

Hardware problems: 15%
Software problems: 34%
Operator error: 51%

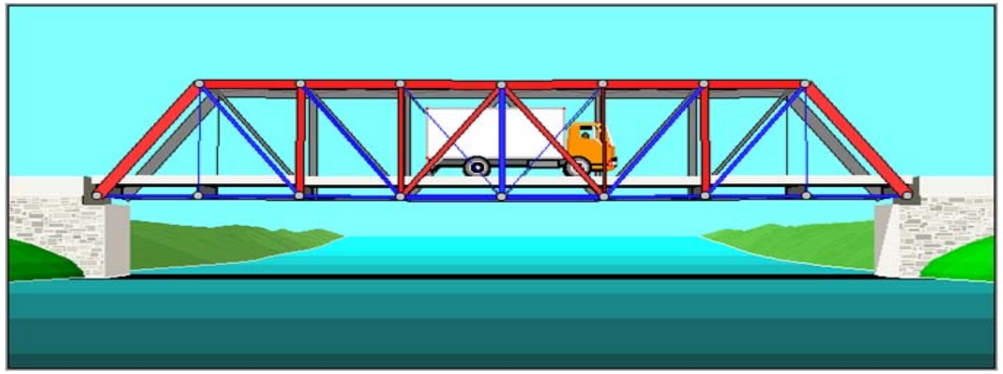
[1] Definition used by the Tech. Council on Forensic Engineering of the Amer. Society of Civil Engineers

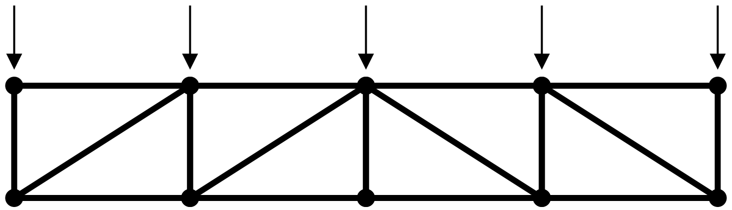

How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Design Flaws: "To Engineer is Human"[1]

Complex systems almost certainly contain multiple design flaws

Redundancy in the form of safety factor is routinely used in buildings and bridges





Example of a more subtle flaw: Disney Concert Hall in Los Angeles reflected light into nearby building, causing discomfort for tenants due to blinding light and high temperature

[1] Title of book by Henry Petroski

Yet Another Example

The Walkie Talkie melted my Jag! Light reflected from under-construction City skyscraper buckles bodywork and mirror of businessman's car

- **Sunlight reflected from skyscraper is causing heat damage to cars beneath**
- **Several panels of a Jaguar XJ had buckled in the glare**
- **Other drivers say their vehicles have wilted in the beam of light**

By SAM WEBB

PUBLISHED: 18:56 GMT, 2 September 2013 | **UPDATED:** 22:39 GMT, 3 September 2013

A £200million skyscraper has had an undeniably dazzling effect on passers-by – but not, unfortunately, the one intended by its architect.

Learning Curve: "Normal Accidents"[1]

Example: Risk of piloting a plane

1903 First powered flight

1908 First fatal accident

1910 Fatalities = 32 (≈2000 pilots worldwide)

1918 US Air Mail Service founded
Pilot life expectancy = 4 years
31 of the first 40 pilots died in service

1922 One forced landing for every 20 hours of flight

Today Commercial airline pilots pay normal life insurance rates

Unfortunately, the learning curve for computers and computer-based systems is not as impressive

[1] Title of book by Charles Perrow (Ex. p. 125)

Mishaps, Accidents, and Catastrophes

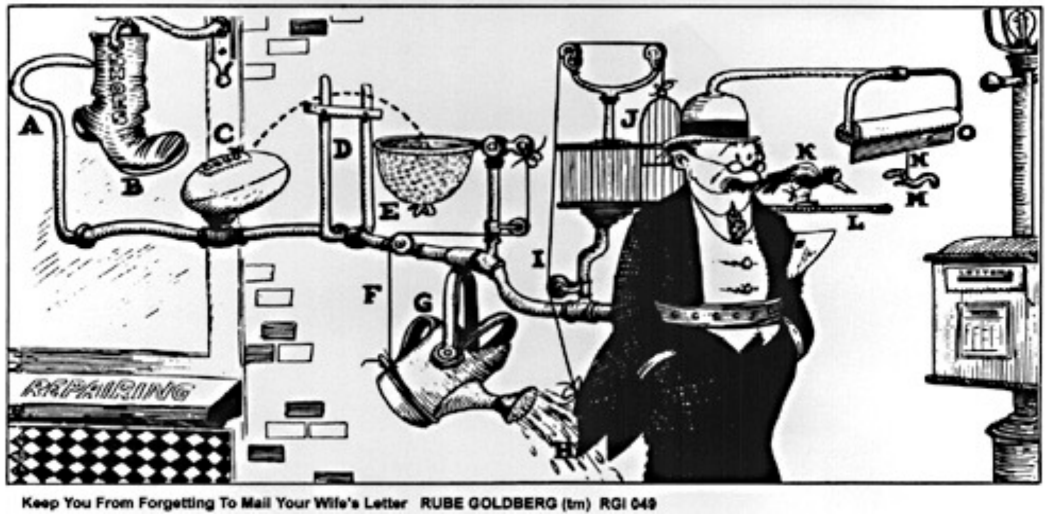
Mishap: misfortune; unfortunate accident

Accident: unexpected (no-fault) happening causing loss or injury

Catastrophe: final, momentous event of drastic action; utter failure

At one time (following the initial years of highly unreliable hardware), computer mishaps were predominantly the results of human error

Now, most mishaps are due to complexity (unanticipated interactions)



Rube Goldberg contraptions

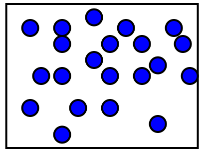
The butterfly effect

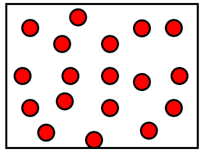
Pretest: Failures and Probabilities

This test will not be graded or even collected, so answer the test questions truthfully and to the best of your ability / knowledge

Question 1: Name a disaster that was caused by computer hardware or software failure. How do you define “disaster” and “failure”?

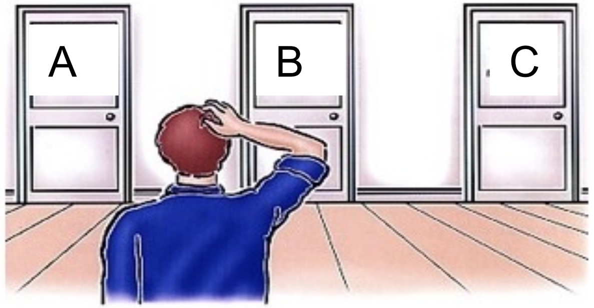
Question 2: Which of these patterns is more random?





Question 3: Which do you think is more likely: the event that everyone in this class was born in the first half of the year or the event that at least two people were born on the same day of the year?

Question 4: In a game show, there is a prize behind one of 3 doors with equal probabilities. You pick Door A. The host opens Door B to reveal that there is no prize behind it. The host then gives you a chance to switch to Door C.
Is it better to switch or to stick to your choice?



Pretest (Continued): Causes of Mishaps

Question 5: Does this photo depict a mishap due to design flaw, implementation bug, procedural inadequacies, or human error?

Pretest (Continued): Reliability and Risk

Question 6: Name an emergency backup system (something not normally used unless another system fails) that is quite commonplace

Question 7: Which is more reliable: plane X or plane Y that carries four times as many passengers as plane X and is twice as likely to crash?

Question 8: Which is more reliable: a 4-wheel vehicle with one spare tire or an 18-wheeler with 2 spare tires?

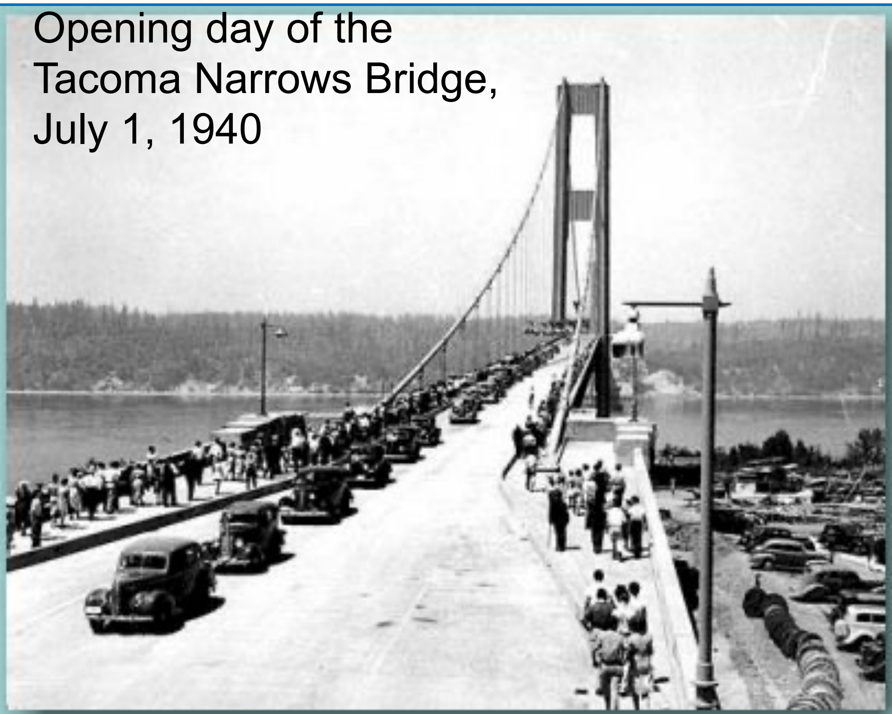
Question 9: Which surgeon would you prefer for an operation that you must undergo: Surgeon A, who has performed some 500 operations of the same type, with 5 of his patients perishing during or immediately after surgery, or surgeon B who has a perfect record in 25 operations?

Question 10: Which is more probable at your home or office: a power failure or an Internet outage? Which is likely to last longer?

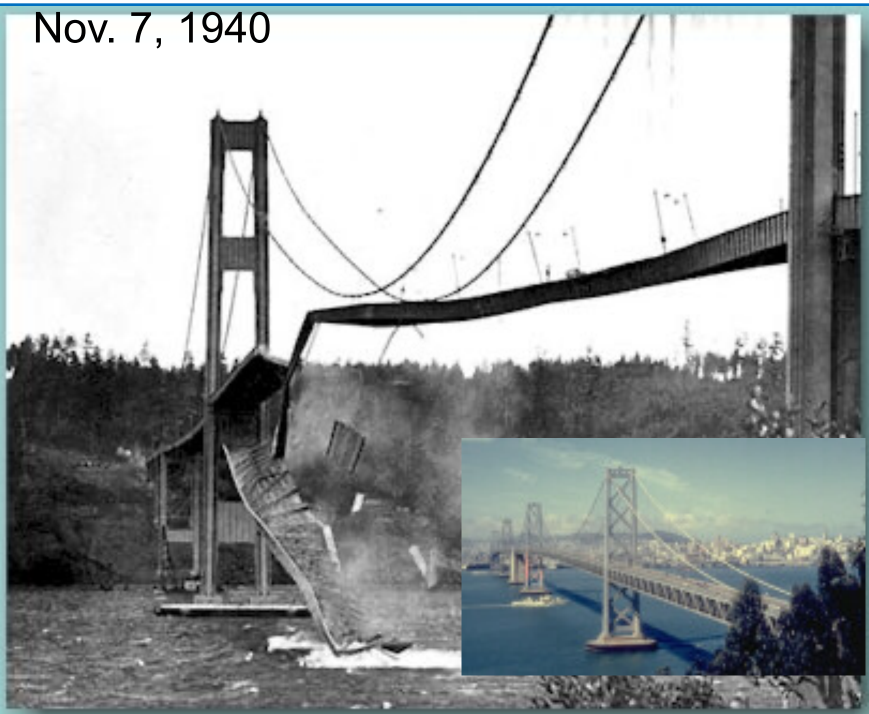
If you had trouble with 3 or more questions, you really need this course!

What Do We Learn from Bridges that Collapse?

Opening day of the Tacoma Narrows Bridge, July 1, 1940



Nov. 7, 1940



One catastrophic bridge collapse every 30 years or so

See the following amazing video clip (Tacoma Narrows Bridge):
http://www.enm.bris.ac.uk/research/nonlinear/tacoma/tacnarr.mpg

" . . . failures appear to be inevitable in the wake of prolonged success, which encourages lower margins of safety. Failures in turn lead to greater safety margins and, hence, new periods of success."

Henry Petroski, *To Engineer is Human*

. . . or from "Unsinkable" Ships that Sink?

"The major difference between a thing that might go wrong and a thing that cannot possibly go wrong is that when a thing that cannot possibly go wrong goes wrong, it usually turns out to be impossible to get at or repair."

Douglas Adams, author of *The Hitchhiker's Guide to the Galaxy*

… or from Poorly Designed High-Tech Trains?

Train built for demonstrating magnetic levitation technology in northwest Germany rams into maintenance vehicle left on track at 200 km/h, killing 23 of 29 aboard

Official investigation blames the accident on human error (train was allowed to depart before a clearance phone call from maintenance crew)

Not a good explanation; even low-tech trains have obstacle detection systems

Even if manual protocol is fully adequate under normal conditions, any engineering design must take unusual circumstances into account (abuse, sabotage, terrorism)

Design Flaws in Computer Systems

Hardware example: Intel Pentium processor, 1994

For certain operands, the FDIV instruction yielded a wrong quotient
Amply documented and reasons well-known (overzealous optimization)

Software example: Patriot missile guidance, 1991

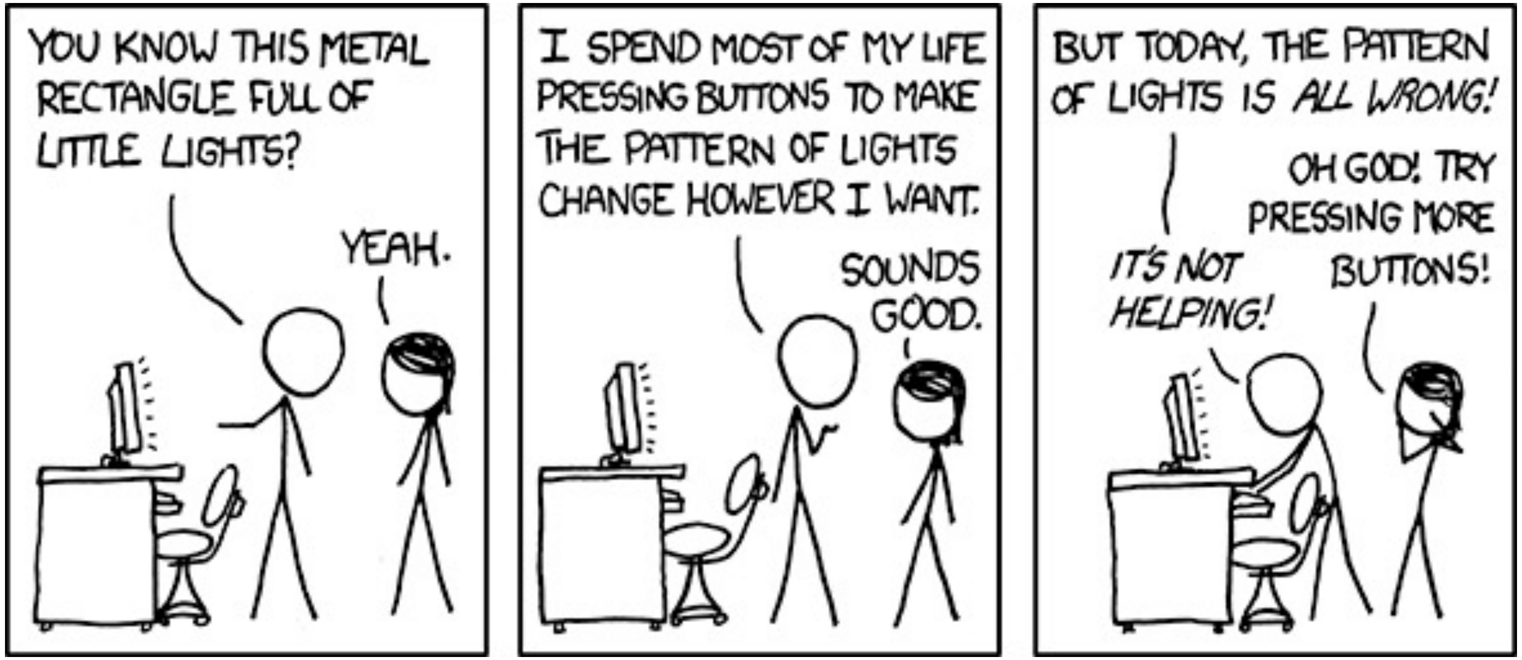
Missed intercepting a scud missile in 1st Gulf War, causing 28 deaths
Clock reading multiplied by 24-bit representation of 1/10 s (unit of time) caused an error of about 0.0001%; normally, this would cancel out in relative time calculations, but owing to ad hoc updates to some (not all) calls to a routine, calculated time was off by 0.34 s (over $\approx$100 hours), during which time a scud missile travels more than ½ km

User interface example: Therac 25 machine, mid 1980s[1]

Serious burns and some deaths due to overdose in radiation therapy
Operator entered "x" (for x-ray), realized error, corrected by entering "e" (for low-power electron beam) before activating the machine; activation was so quick that software had not yet processed the override

[1] Accounts of the reasons vary

XKCD of the Day



http://xkcd.com/722/

Causes of Human Errors in Computer Systems

1. Personal factors (35%): Lack of skill, lack of interest or motivation, fatigue, poor memory, age or disability

2. System design (20%): Insufficient time for reaction, tedium, lack of incentive for accuracy, inconsistent requirements or formats

3. Written instructions (10%): Hard to understand, incomplete or inaccurate, not up to date, poorly organized

4. Training (10%): Insufficient, not customized to needs, not up to date

5. Human-computer interface (10%): Poor display quality, fonts used, need to remember long codes, ergonomic factors

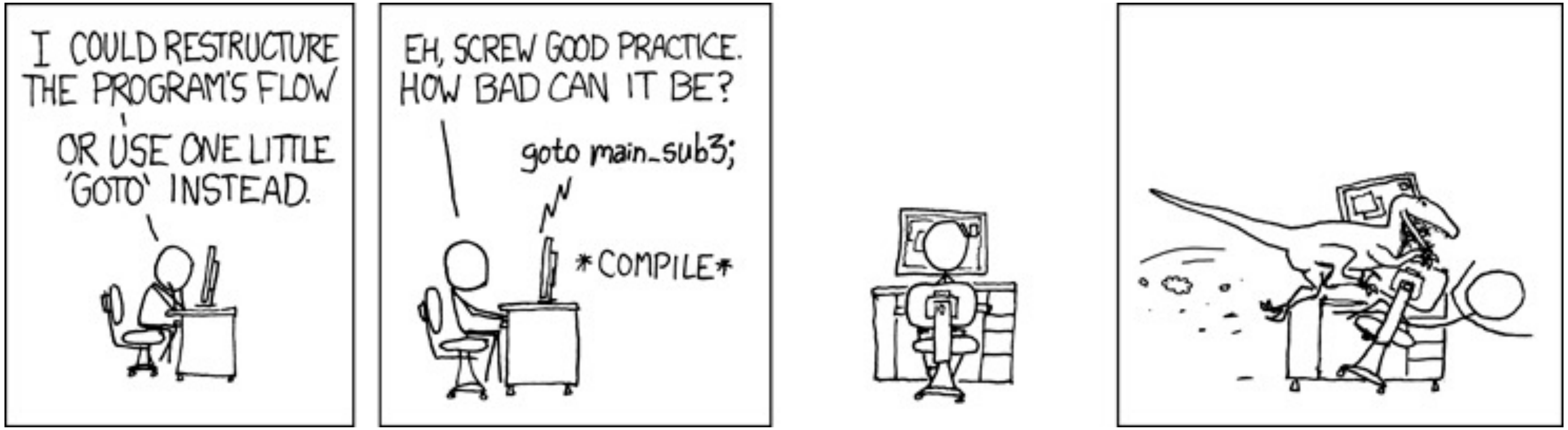
6. Accuracy requirements (10%): Too much expected of operator

7. Environment (5%): Lighting, temperature, humidity, noise

Because "the interface is the system" (according to a popular saying), items 2, 5, and 6 (40%) could be categorized under user interface


NO! - Bad User!!!
You've been warned 3 times that this file does not exist.
Now you've made us catch this worthless exception and we're upset.
Do not do this again.
OK


Big error
You really screwed up this time.
I know
What else is new
It's my first day



http://xkcd.com/292/

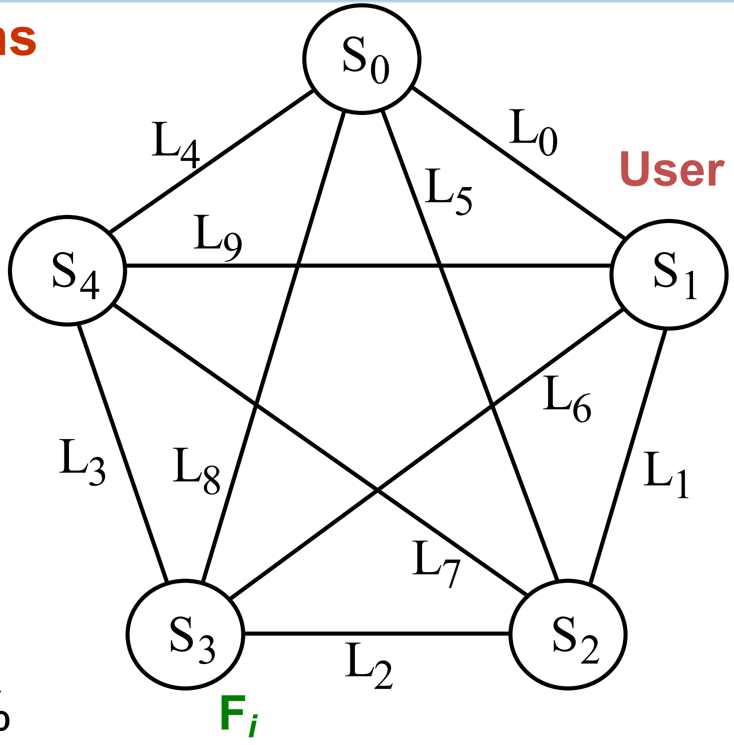
A Case Study

Data availability and integrity concerns

Distributed DB system with 5 sites
Full connectivity, dedicated links
Only direct communication allowed
Sites and links may malfunction
Redundancy improves availability

S: Probability of a site being available
L: Probability of a link being available

Single-copy availability = SL
Unavailability = $1 - SL$
= $1 - 0.99 \times 0.95 = 5.95\%$

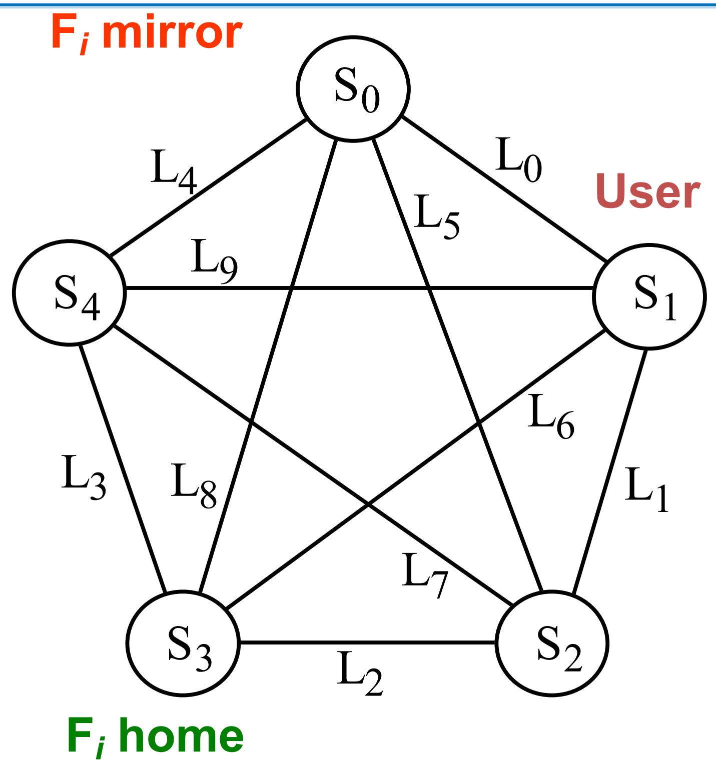


Data replication methods, and a challenge

File duplication: home / mirror sites
File triplication: home / backup 1 / backup 2
Are there availability improvement methods with less redundancy?

Data Duplication: Home and Mirror Sites

S: Site availability e.g., 99%
L: Link availability e.g., 95%



$$A = SL + (1 - SL)SL$$

- SL: Primary site can be reached
- $(1 - SL)$: Primary site inaccessible
- SL: Mirror site can be reached

Duplicated availability = $2SL - (SL)^2$
Unavailability = $1 - 2SL + (SL)^2$
= $(1 - SL)^2$ = 0.35%

Data unavailability reduced from 5.95% to 0.35%

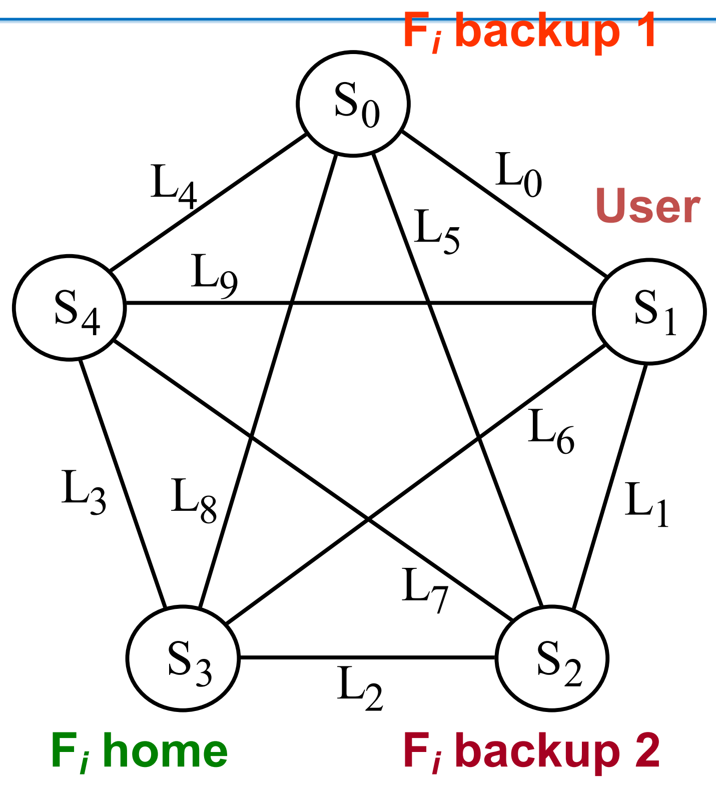
Availability improved from ≈ 94% to 99.65%

Data Triplication: Home and Two Backups

S: Site availability e.g., 99%
L: Link availability e.g., 95%

$A = SL + (1 - SL)SL + (1 - SL)^2SL$

- SL — Primary site can be reached
- $(1 - SL)$ — Primary site inaccessible
- SL — Backup 1 can be reached
- $(1 - SL)^2$ — Primary and backup 1 inaccessible
- SL — Backup 2 can be reached

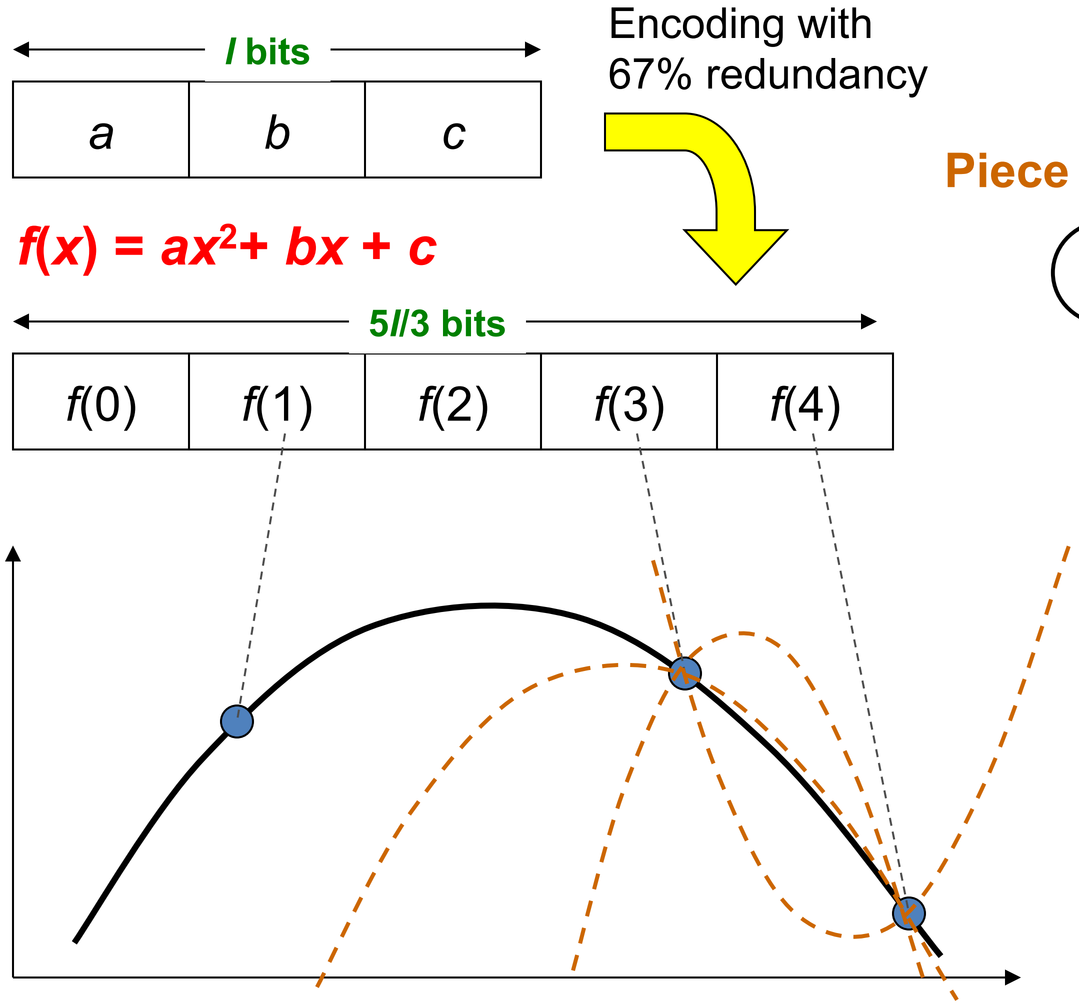


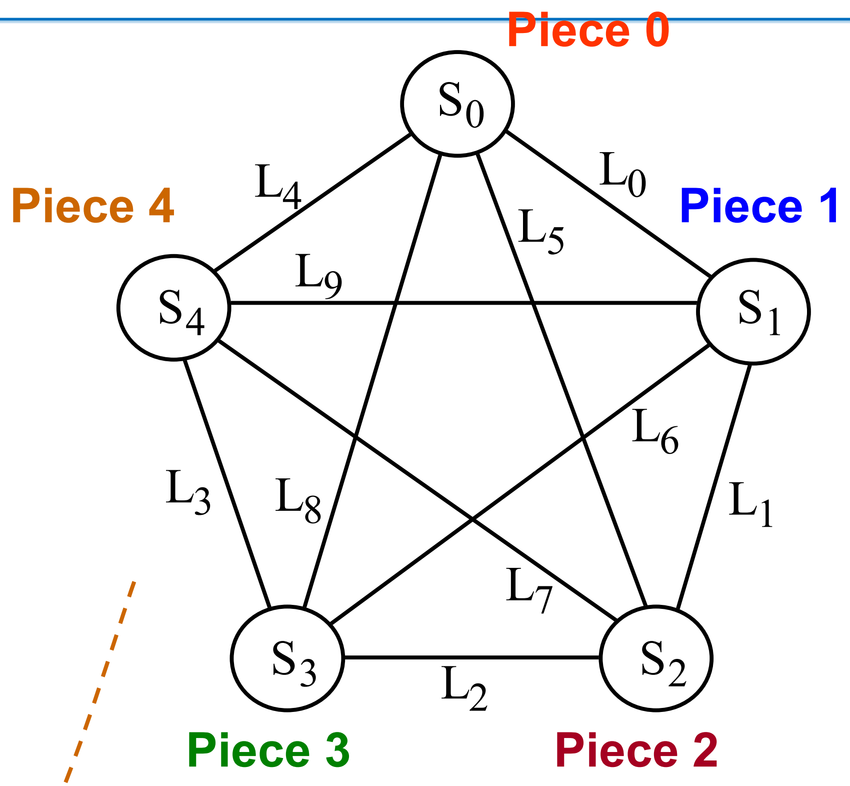
Triplicated avail. $= 3SL - 3(SL)^2 - (SL)^3$
Unavailability $= 1 - 3SL - 3(SL)^2 + (SL)^3$
$= (1 - SL)^3 = 0.02\%$

Data unavailability reduced from 5.95% to 0.02%

Availability improved from ≈ 94% to 99.98%

Dispersion for Data Security and Integrity





Note that two pieces would be inadequate for reconstruction

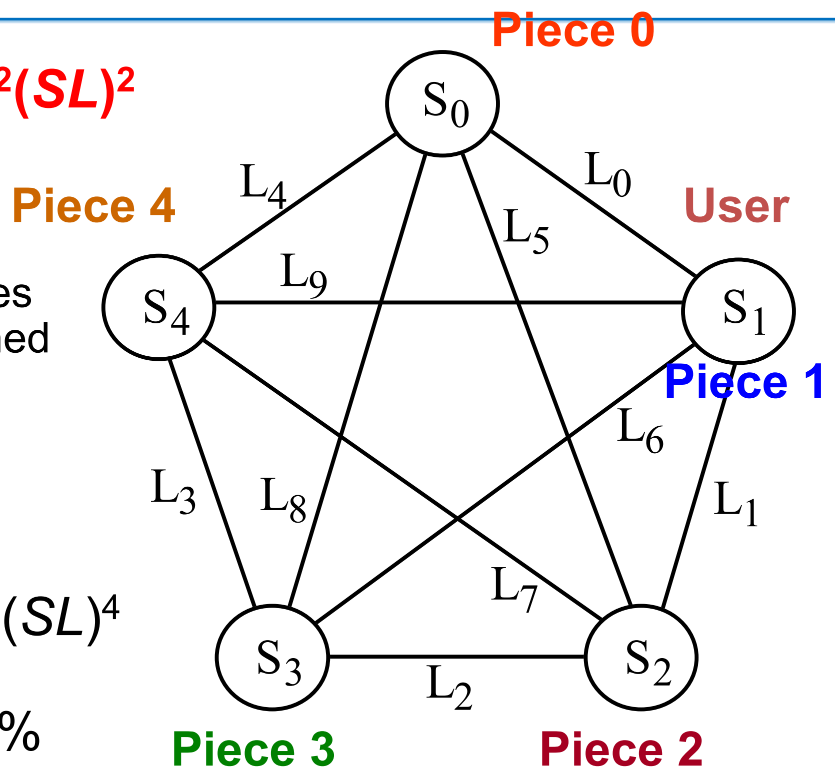
Data Dispersion: Three of Five Pieces

$A = (SL)^4 + 4(1 - SL)(SL)^3 + 6(1 - SL)^2(SL)^2$

All 4 pieces can be reached	Exactly 3 pieces can be reached	Only 2 pieces can be reached

S: Site availability e.g., 99%
L: Link availability e.g., 95%

Dispersed avail. = $6(SL)^2 - 8(SL)^3 + 3(SL)^4$
Availability = 99.92%
Unavailability = 1 – Availability = 0.08%



Scheme →	Nonredund.	Duplication	Triplication	Dispersion
Unavailability	5.95%	0.35%	0.02%	0.08%
Redundancy	0%	100%	200%	67%

Questions Ignored in Our Simple Example

1. How redundant copies of data are kept consistent
When a user modifies the data, how to update the redundant copies (pieces) quickly and prevent the use of stale data in the meantime?

2. How malfunctioning sites and links are identified
Malfunction diagnosis must be quick to avoid data contamination

3. How recovery is accomplished when a malfunctioning site / link returns to service after repair
The returning site must be brought up to date with regard to changes

4. How data corrupted by the actions of an adversary is detected
This is more difficult than detecting random malfunctions

The example does demonstrate, however, that:

- Many alternatives are available for improving dependability
- Proposed methods must be assessed through modeling
- The most cost-effective solution may be far from obvious

Acknowledgements

- Aceste slide-uri contin materiale dezvoltate de:

  - Behrooz Parhami, UCSB, ECE257A