- Exams: July 18 and September 4 (5)
- watch out for “Systems Programming Lab” in Fall !!!
MODELING DISTRIBUTED SYSTEMS

HERMANN HÄRTIG, DISTRIBUTED OPERATING SYSTEMS, SS2017
MODELS IN GENERAL

- abstract from details
- concentrate on functionality, properties, ... that are considered important for a specific system/application
- use model to analyze, prove, predict, ... system properties and to establish fundamental insights
- models in engineering disciplines very common, not (yet) so in CS
- we'll see many models in “Real-Time Systems” class
Reasoning:
- Common sense
- Formal Verification
- Careful Inspection
- Mathematics

THE GENERAL APPROACH

Modeling Distributed Systems
THE GENERAL APPROACH

Reasoning:
- Common sense
- Formal Verification
- Careful Inspection
- Mathematics

“Refinement”:
- Abstraction
- Implementation
- Formal Refinement

Modeling Distributed Systems
## MODEL EXAMPLES IN GENERAL

<table>
<thead>
<tr>
<th>Model</th>
<th>Objective/Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Trees</td>
<td>are all failure combinations taken into account</td>
</tr>
<tr>
<td>static models</td>
<td>does a house eventually fall down</td>
</tr>
<tr>
<td></td>
<td>what kind of vehicles on a bridge</td>
</tr>
<tr>
<td>control laws</td>
<td>stability of controllers</td>
</tr>
<tr>
<td>Ohm’s Law</td>
<td>behavior of circuits</td>
</tr>
</tbody>
</table>
WELL KNOWN EXAMPLES FOR MODELS

\[ I = \frac{V}{R} \]
<table>
<thead>
<tr>
<th>Model</th>
<th>Objective/Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turing Machine</td>
<td>Decidability</td>
</tr>
<tr>
<td>Amdahl’s Law</td>
<td>Scalability</td>
</tr>
<tr>
<td>Logic</td>
<td>Correctness, Precision, …</td>
</tr>
<tr>
<td>Real-Time “tasks”</td>
<td>can all timing requirements be met</td>
</tr>
<tr>
<td>Byzantine Agreement</td>
<td>Consensus</td>
</tr>
<tr>
<td>Two Army</td>
<td>Consensus</td>
</tr>
</tbody>
</table>
UML ???
Objective of lecture:
understand the power of models and the need for their careful understanding

Intuition, No (real) proofs
Q1: Is possible to build arbitrarily reliable Systems out of unreliable components?

Q2: Can we achieve consensus in the presence of faults (consensus: all non-faulty components agree on action)?

Q3: Is there an algorithm to determine for a system with a given setting of access control permissions, whether or not a Subject A can obtain a right on Object B?

2 Models per Question!

All questions/answers/models -> published 1956 - 1982 !!!
Q1: Can we build arbitrarily reliable Systems out of unreliable components?

- How to build reliable systems from less reliable components
- Fault(Errors, Failure, Fault, ....) terminology in this lecture synonymously used for “something goes wrong” (more precise definitions and types of faults in SE)
DEFINITIONS

Reliability:

- $R(t)$: probability for a system to survive time $t$

Availability:

- $A$: fraction of time a system works
LIST OF INGREDIENTS

- Fault detection and confinement
- Recovery
- Repair
- Redundancy
  - Information
  - time
  - structural
  - functional
John v. Neumann
Voter: single point of failure

Can we do better
→ distributed solutions?
Parallel-Serial-Systems

(Pfitzmann/Härtig 1982)
Parallel-Serial-Systems
(Pfitzmann/Härtig 1982)
Parallel-Serial-Systems
(Pfitzmann/Härtig 1982)
Parallel-Serial-Systems
(Pfitzmann/Härtig 1982)
Serial-Systems

Each component must work for the whole system to work.

$$R_{\text{whole}} = \prod_{j=1}^{m} R_j$$
Parallel-Systems

One component must work for the whole system to work. Each component must fail for the whole system to fail.

\[ R_{\text{whole}} = 1 - \prod_{i=1}^{m} (1 - R_i) \]
Q1/MODEL1: ABSTRACT MODEL

Serial-Parallel-Systems

\[ R_{whole} = 1 - \prod_{j=1}^{m} \left( 1 - \prod_{i=1}^{n} R_{i,j} \right) \]
Fault Model

„Computer-Bus-Connector” can fail such that Computer and/or Bus also fail

=>

conceptual separation of components into

- Computer, Bus: can fail per se
- CC: Computer-Connector fault also breaks the Computer
- BC: Bus-Connector fault also breaks Bus
Q1/MODEL1: CONCRETE MODEL

1 Buses

1 Computers

Computer 1

Bus 1

CC 1,1

BC 1,1
Q1/MODEL1: CONCRETE MODEL

1 Buses
2 Computers

- Computer 1
  - Bus 1
    - BC 1,1
    - BC 2,1

- Computer 2
  - CC 2,1

Bus 1

1 Buses
2 Computers
Q1/Model1: Concrete Model

1 Buses
N Computers

Bus 1
- BC 1,1
- BC 2,1
- BC n,1

Computer 1
- CC 1,1

Computer 2
- CC 2,1

Computer n
- CC n,1
Q1/MODEL1: CONCRETE MODEL

M Buses
1 Computers

Bus 1
- BC 1,1

Bus m
- BC 1,m

Computer 1
- CC 1,1
- CC 1,2
- CC 1,m

M Buses
1 Computers
Q1/MODEL1: CONCRETE MODEL

M Buses
N Computers
Q1/MODEL1: CONCRETE MODEL FOR N,M

\[ R_{whole}(n, m) = \left( 1 - \left( 1 - R_{Bus} \cdot R_{BC}^n \right)^m \right) \left( 1 - \left( 1 - R_{Computer} \cdot R_{CC}^m \right)^n \right) \]

\[ \text{then: } R_{CC}, R_{BC} < 1 : \lim_{n, m \to \infty} R(n, m) = \]
System built of Synapses (John von Neumann, 1956)

Computation and Fault Model:
- Synapses deliver "0" or "1"
- Synapses deliver with $R > 0.5$:
  - with probability $R$ correct result
  - with $(1-R)$ wrong result

Then we can build systems that deliver correct result for any (arbitrarily high) probability $R$
Q2: Can we achieve consensus in the presence of faults all non-faulty components agree on action?

- all correctly working units agree on result/action
- agreement non trivial (based on exchange of messages)
Q2/Model 1: "2 Army Problem"

- p,q processes
  - communicate using messages
  - messages can get lost
  - no upper time for message delivery known
  - do not crash, do not cheat

- p,q to agree on action (e.g. attack, retreat, ...)

- how many messages needed?

- first mentioned: Jim Gray 1978
Result: there is no protocol with finite messages

Prove by contradiction:

- assume there are finite protocols (mp --> q, mq --> p)*
- choose the shortest protocol MP,
- last message MX: mp --> q or mq --> p
- MX can get lost
- => must not be relied upon => can be omitted
- => MP not the shortest protocol.
- => no finite protocol
n processes, f traitors, n-f loyals

- communicate by reliable and timely messages (synchronous messages)
- traitors lye, also cheat on forwarding messages
- try to confuse loyals
Goal:

- loyals try to agree on non-trivial action (attack, retreat)
- non-trivial more specific:
  - one process is commander
  - if commander is loyal and gives an order, loyals follow the order otherwise loyals agree on arbitrary action
Q2/MODEL 2: “BYZANTINE AGREEMENT”

3 Processes: 1 traitor, 2 loyals

Commander

Lieutenant

Lieutenant

attack

he said: retreat

attack
Q2/MODEL 2: “BYZANTINE AGREEMENT”

3 Processes: 1 traitor, 2 loyals

=> 3 processes not sufficient to tolerate 1 traitor

He said: retreat
Q2/Model 2: "Byzantine Agreement"

4 Processes

- Commander
- Lieutenant
- Lieutenant
- Lieutenant

- Attack from Commander to Lieutenant 1
- Attack from Lieutenant to Commander
- Attack from Lieutenant to Lieutenant
- Attack from Lieutenant to Lieutenant

He said:
- Attack from Lieutenant to Lieutenant
- Retreat from Lieutenant to Lieutenant
4 Processes

all lieutenant receive x, y, z => can decide

**General result:** 3 f + 1 processes needed to tolerate f traitors
Q3: Is there an algorithm to determine for a system with a given setting of access control permissions, whether or not a Subject A can obtain a right on Object B?