Introduction

• System availability/reliability/dependability
  = “ability of system to provide ‘correct’ (specified) functionality”
  – for communication networks, e.g.
    • Connectivity
    • Services (network services, applications)
• Properties when applied to complex systems:
  – Specification of ‘correct’ functionality difficult (as well as verification and testing)
  – Stateful systems: individual faults lead to subsequent dependent faults (⇒ significance of fault detection)
  – Dependability is in general load dependent; faults influence performance
    ⇒ strong mutual relation between performance and dependability aspects (‘performability’)
• Analysis of dependability aspects via
  – Analytic models: Frequently independence and Markovian assumptions
  – Simulation models: Relevance of rare-event techniques
  – Experimental Settings (Prototype/Test systems)
Main Application Areas

- Safety critical systems
  - Air-plane control systems
  - Vehicular electronic systems (e.g. ABS)
- Satellites/Space missions
- Business critical systems
  - Electronic stock markets
  - Electronic ordering/booking systems
- High-Performance/Parallel Computing
- Distributed Systems: Telecommunication systems
  - Traditional PSTNs: Availability 99.94% (end-to-end)
  - Internet: below 99%

Motivation: Failure Types in Communication NWs

- IP networks designed to be ‘robust’ but not highly available
- End-to-end availability
  - Public Internet: below 99% (see e.g. [http://www.netmon.com/IRI.asp](http://www.netmon.com/IRI.asp)),
  - PSTN: 99.94%
- Operation, Administration, and Maintenance (OAM) one major source of outage times
Content

1. Motivation and basic definitions

2. Mathematical models
   a) Availability models
   b) Systems without repair: Life-time models
   c) Systems with Repair

3. Protocols for fault-tolerant networks/services
   a) SW Fault-Tolerance [opt.]
   b) Network Availability
      • IP Routing, HSRP/VRRP, SCTP
   c) Server/Service Availability
      • Server Farms, Cluster solution, distributed reliability

4. Summary/Exercises

Basic concepts (I)

• Fault/defect/error: component behavior deviates from desired behavior
  – Transient vs. Persistent faults
  – Possible causes:
    • Design errors
    • Aging/wear-out
    • Mis-use
    • Maintenance (planned down-times)

• System Failure
  – Observed system behavior deviates from specification, defined by
  – Failure criteria (time/functionality)

• Note: Faults are a necessary condition for failures but not sufficient
  – Fault-tolerance, 'robust' failure criteria

• Redundancy
  – Active (functional) vs passive (non-functional, stand-by)
  – Structural (components) vs. Information (e.g. additional header fields) vs. Temporal (retries)
Basic concepts (II): Metrics

- **Availability (‘accessibility’) A(t)**
  - System operational at time t
  - System accepts a service request at time t

- **Reliability R(t₁,t₂)**
  - System operational within time interval [t₁,t₂] provided it was operational (‘accessible’) at time t₁
  - System serves a request given that it had accepted it at time t₁

- **Dependability D(t₁,t₂)**
  - System available at t₁ and operational within time interval [t₁,t₂]
  - Successfully accepts and serves a service request

- **Systems without repair**
  - Distribution of Life-times
  - Hazard/failure rates

- **Systems with repair**
  - Distribution/Mean of time to failure (‘up-time’) 
  - Distribution/Mean of time to repair (‘down-time’)

---

**Content**

1. Motivation and basic definitions

2. Mathematical models
   a) Availability models
   b) Systems without repair: Life-time models
   c) Systems with Repair

3. Protocols for fault-tolerant networks/services
   a) SW Fault-Tolerance [opt.]
   b) Network Availability
      • IP Routing, HSRP/VRP, SCTP
   c) Server/Service Availability
      • Server Farms, Cluster solution, distributed reliability

4. Summary/Exercises
Availability Models (I)

- System description
  - System consists of N components
  - Components states $Z_1, ..., Z_N \rightarrow$ availability state vector
    - Binary: $Z_i$ in $\{0, 1\}$
    - Multi-state or continuous
  - System availability: function of component availability $Z=f(Z_1, ..., Z_N)$
    - Disjunct normal form (DNF) of binary availability function
    - Monotonicity, coherence (no irrelevant components)
- Availability graph (binary case)
  - Graph: Start node S, End-node E, Path for each 'summand' in DNF, node for each 'factor'
  - Representation of system availability function
  - Simplification of graph through factorization

Availability Models (II)

Common Patterns (with independence assumptions!)

- Serial structures
  - All components need to be available for system availability
    - $A=\prod A_i$
  - Boolescher Term and Availability Graph:

- Parallel structures
  - At least one component has to be available for system availability
    - $A=1-\prod (1-A_i)$
  - Boolescher Term and Availability Graph:

- k-out-of-n systems
  - At least k out of n identical components have to be available for system availability
  - Use-cases: load-balancing/throughput, failure detection by majority votes
    - $A=\sum_{i=k}^{n} \binom{n}{i} A_i(1-A_i)^{n-i}$
  - Special cases: Parallel structure (k=1) and serial structure (k=n)
- General case: $A=...$
Availability Models (III): Examples

Examples
- Triple Modular Redundancy (TMR)
  - 3 functionally identical components, result by majority vote
    2-out-of-3 case
- Efficient use of redundancy in serial structure?
  - Replication of serial structure
  - Replicate each component
  - Combinations
  when neglecting components necessary for
    - Failure detection
    - Node Elimination
    - Majority votes

Systems without Repair: Lifetime models (I)
- States time-dependent, $Z_i(t)$ and $Z(t) \rightarrow$ description as stochastic process
- Time $LT = \min_i \{Z(t)=0\}$ is called life-time of system
  - Compare to first passage times in queueing models
- Reliability: $R(t)=Pr(Z(t)=1)$ [for systems without repair]
  - Serial structure: $R(t)=\prod R_i(t)$ [independence]
    - Example: exponential distributed LTs $\rightarrow$ $E[LT]=1/(\Sigma \lambda_i)$
  - Parallel structure: $R(t)= 1- \prod (1-R_i(t))$
    - Example: exponential distributed LTs $\rightarrow$ $E[LT]=1/\Sigma (1/i)$ (law of diminishing return)
- Hazard/failure Rate:
  - $\lambda(t)=\lim_{\Delta \to 0} 1/\Delta Pr(\text{failure occurs in interval } [t;t+\Delta] \mid \text{system operational at time } t')$
    $= f_{LT}(t)/R(t)$ (with $f_{LT}(t)$ pdf of LT)
  - Constant Hazard rate $\Leftrightarrow$ exponential life-time
Lifetime models (II)

- Typical hazard rates
  - HW: bath-tub functions
  - Exponential LT
  - Power-tailed LT
- Examples:
  - Exponentially distributed lifetimes of components
    → Hazard Rate of serial structure equals sum of individual rates
  - TMR
    - $R(t)$, $E(LT)$, condition for $R_{TMR}(t) > R(t)$

Systems with Repair

- Description of component failure and repair processes
  - Repair time distribution with mean MTTR
  - Failure Time distribution with mean MTTF
    Common assumption for modeling: Exponential repair time and exponential time to failure
- Availability: $A(t) = \text{Pr('system operational at time $t$')}$
  - $\lim A(t) = \text{MTTF}/(\text{MTTR} + \text{MTTF})$
- Reliability: $R(t, t+\Delta) = \text{Pr('system operational in $[t, t+\Delta]$ | operational at time $t$')}$
- Dependability: $D(t, t+\Delta) = \text{Pr('system operational in $[t, t+\Delta]$')}$
  - $D(t, t+\Delta) = R(t, t+\Delta) A(t)$ (definition of conditional probabilities)
- Example: Markovian models for
  - Single back-up component (passive stand-by): stationary probabilities
    - Redundant unit fails only when in operation
    - Redundant unit can also fail in stand-by
    - TMR
Content

1. Motivation and basic definitions
2. Mathematical models
   a) Availability models
   b) Systems without repair: Life-time models
   c) Systems with Repair
3. Protocols for fault-tolerant networks/services
   a) SW Fault-Tolerance [opt.]
   b) Network Availability
      • IP Routing, HSRP/VRRP, SCTP
   c) Server/Service Availability
      • Server Farms, Cluster solution, distributed reliability
4. Summary/Exercises

General Approaches: Fault-tolerance

• Basic requirements for fault-tolerance
  – Number of fault-types and number of faults is bounded
  – Existence of redundancy (structural: equal/diversity; functional; information; time-redundancy/retries)
• Functional Parts of fault-tolerant systems
  – Fault detection (& diagnosis)
    • Replication and Comparison (if identical realisations → not suitable for design errors!)
    • Inversion (e.g. Mathematical functions)
    • Acceptance (necessary conditions on result, e.g. ranges)
    • Timing behavior (time-outs)
  – Fault isolation: prevent spreading
    • Isolation of functional components, e.g. Atomic actions, layering model
  – Fault recovery
    • Backward: Retry, Journalling (Restart), Checkpointing & Rollback (non-trivial in distributed cooperating systems!)
    • Forward: move to consistent, acceptable, safe new state; but loss of result
    • Compensation, e.g. TMR, FEC
**Software fault-tolerance [opt.]**

- Mainly: Design errors & user interaction (as opposed to production errors, wear-out, etc.)
- Observations/Estimates (experience in computing centers, numbers a bit old however)
  - 0.25...10 errors per 1000 lines of code
  - Only about 30% of error reports by users accepted as errors by vendor
  - Reaction times (updates/patches): weeks to months
  - Reliability not nearly as improved as hardware errors, various reasons:
    - immensely increased software complexity/size
    - Faster HW → more operations per time-unit executed → higher hazard rate
  - However, relatively short down-times (as opposed to HW): ca 25 min

- Approaches for fault-tolerance
  - Software diversity (n-version programming, back-to-back tests of components)
    - However, same specification often leads to similar errors
    - Forced diversity as improvement (?)
    - Majority votes for complex result types not trivial

**Network Connectivity: Basic Resilience**

- Loss of network connectivity
  - Link Failures (cable cut)
  - Router/component failures along the path
- Basic resilience features on (almost) every protocol layer
  - L1+L2: FEC, Link-Layer Retransmission, Resilient Switching (e.g. Resilient Packet Ring)
  - L3, IP: Dynamic Routing
  - TCP: Retransmissions
  - Application: application-level retransmissions, loss-resilient coding (e.g. VoIP, Video)

- IP Layer Network Resilience: Dynamic Routing, e.g. OSPF
  - ‘Hello’ packets used to determine adjacencies and link-states
  - Missing hello packets (typically 3) indicate outages of links or routers
  - Link-states propagated through Link-state advertisements (LSA)
  - Updated link-state information (adjacencies) lead to modified path selection
**Dynamic Routing: Improvements**

- Drawbacks of dynamic routing
  - Long Duration until newly converged routing tables (30sec up to several minutes)
  - Rerouting not possible if first router (gateway) fails
- Improvements
  - Speed-up: Pre-determined secondary paths (e.g. Via MPLS)
  - 'Local' router redundancy:
    - Hot Standby (HSRP, RFC2281) & Virtual Router Redundancy Protocol (VRRP, RFC2338)
      - Multiple routers on same LAN
      - Master performs packet routing
      - Fail-over by migration of 'virtual' MAC address

**Streaming Control Transmission Protocol (SCTP)**

- Defined in RFC2960 (see also RFC 3257, 3286)
- Purpose initially: Signalling Transport
- Features
  - Reliable, full-duplex unicast transport (performs retransmissions)
  - TCP-friendly flow control (+ many other features of TCP)
  - Multi-streaming, in sequence delivery within streams
    - Avoid head of line blocking (performance issue)
  - Multi-homing: hosts with multiple IP addresses, path monitoring (heart-beat mechanism), transparent failover to secondary paths
    - Useful for provisioning of network reliability
Content

1. Motivation and basic definitions
2. Mathematical models
   a) Availability models
   b) Systems without repair: Life-time models
   c) Systems with Repair
3. Protocols for fault-tolerant networks/services
   a) SW Fault-Tolerance (opt.)
   b) Network Availability
      • IP Routing, HSRP/VRRP, SCTP
   c) Server/Service Availability
      • Server Farms, Cluster solution, distributed reliability
4. Summary/Exercises

Resilience for state-less applications

- State-less applications
  - Server response only depends on client request
  - Principle in most IETF protocols, e.g. http
  - If at all, state stored in client, e.g. ‘cookies’
- Redundancy in ‘server farms’ (same LAN)
  - Multiple servers
  - IP address mapped to (replicated) ‘load balancing’ units
    • HW load balancers
    • SW load balancers
  - Load balancers forward requests to specific Server MAC address according to Server Selection Policy, e.g.
    • Random Selection
    • Round-Robin
    • Shortest Response-Time First
- Server failure
  - Load-Balancer does not select failed server any more
- Examples: Web-services, Firewalls
Cluster Framework

- SW Layer across several servers within same subnetwork
- Functionality
  - Load Balancer/dispatching functionality
  - Node Management/
    Node Supervision/Node Elimination
  - Network connection: IP aliasing
    (unsolicited/gratuitous ARP reply)
    \(\Rightarrow\) fail-over times few seconds (up to 10
    minutes, if router/switch does not support
    unsolicited ARP!)
- Single Image view (one IP address for cluster)
  - For communication partners
  - For OAM
- Example Products: Suncluster (SUN), Primecluster (FSC)

Reliability middleware: Example RTP

- Software Layer on top of cluster framework
  - All fault-tolerant operations are done within the cluster
  - One virtual IP address for the whole cluster (UDP dispatcher redirects
    incoming messages) \(\Rightarrow\) “one system image”
  - Transparency for the client
- Redundancy at the level of processes:
  - Each process is replicated among the servers of the cluster
  - Only a faulty process is failed over
- Notion of node disappears:
  - Use of logical name (replicated processes can be
    in the same node)
- Middleware Functionality (Example: Reliable Telco Platform, FSC)
  - Context management
  - Reliable inter-process communication
  - Load Balancing for Network traffic (UDP/SS7 dispatcher)
  - Ticket Manager
  - Events/Alarming
  - Support of rolling upgrade
  - Node Manager
    - Supervision of processes
    - Communication of process status to other
      nodes
  - Trace Manager (Debugging)
    - Administrative Framework (CLI, GUI, SNMP)
**Distributed Redundancy: Reliable Server Pooling**

- Distributed architecture
  - Servers need only IP address
  - Entities offering same service grouped into pools accessible by pool handle
- Pool User (PU) contact servers (PE) after receiving the response to a name resolution request sent to a name server (NS)
  - Name Server monitors PEs
  - Messages for dynamic registration and de-registration
  - Flat Name Space
- Architecture and pool access protocols (ASAP, ENRP) defined in IETF RSerPool WG
- Failure detection and fail-over performed in ASAP layer of client

---

**RSerpool: more details**

- RSerPool Scenario
  - Each PE contains full implementation of functionality, no distributed sub-processes (different to RTP)
  - reduced granularity for possible load balancing
- More Details:
  - RSerPool Name Space
    - Flat Name Space → easier management, performance (no recursive requests)
    - All name servers in operational scope hold same info (about all pools in operational scope)
  - Load Balancing
    - Load factors sent by PE to Name Server (initiated by PE)
    - In resolution requests, Name Server communicates load factors to PU
    - PU can use load factors in selection policies
RSerPool Protocols: Functionality
(Current status in IETF)

- ENRP
  - State-Sharing between Name Servers
- ASAP
  - (De-)Registration of Pool-elements (PE, Name Server)
  - Supervision of Pool-elements by a Name Server
  - Name Resolution (PU, Name Server)
  - PE selection according to policy (& load factors) (PU)
  - Failure detection based on transport layer (e.g. SCTP timeout)
  - Support of Fail-over to other Pool-element (PU-PE)
    - Business cards (pool-to-pool communication)
    - Last will
  - Simple Cookie Mechanism (usable for Pool Element state information) (PU-PE)
  - Under discussion: Application Layer Acknowledgements (PU-PE)

Summary

1. Motivation and basic definitions
2. Mathematical models
   a) Availability models
   b) Systems without repair: Life-time models
   c) Systems with Repair
3. Protocols for fault-tolerant networks/services
   a) SW Fault-Tolerance
   b) Network Availability
      • IP Routing, HSRP/VRRP, SCTP
   c) Server/Service Availability
      • Server Farms, Cluster solution, distributed reliability
4. Summary/Exercises
Topics not treated here

• Dependability and Network Management (OAM)
  – Rolling Upgrade, Patch Concept
  – Hot Plugability/Swapability
  – Graceful Shutdown
  – Resource Supervision/Overload Control, Logging & Tracing
  – Service Concepts/Contracts

• Hardware Platforms
  – Blade Servers

• Performability Models

References/Acknowledgments

• E. Jessen, ‘Dependability and Fault-tolerance of computing systems’, lecture notes (German), TU Munich, 1996.
• Fujitsu Siemens Computers (FSC), ‘Reliable Telco Platform’, Slides.

Related Course:
• ‘Reliable distributed real-time systems’, 9th semester DiRS
Exercises: Availability Models

A service is implemented on three redundant servers in a server farm which is accessed through a HW load balancer, see the network topology below. The individual component availabilities are:

- APs and Routers: A1=99%
- Wired links: A2=99.3%
- Wireless links: A3=85%
- Server Availability (HW+ SW): A4=96%
- HW load balancer: A5=99.8%

Write down the DNF of the service availability functions, draw the availability graphs, and compute the service availabilities for the following two cases:

(I) The service is accessed by a Mobile Node with only a WLAN card

(II) The service is accessed by a multi-homed mobile node that supports a WLAN interface and a Bluetooth interface.

How does the availability in Scenario II change if a distributed (RSerPool like) architecture is deployed, with all 3 servers multi-homed and connected to both routers?