Reliability and Dependability in Computer Networks

CS 552 Computer Networks

Side Credits: A. Tjang, W. Sanders

Outline

- Overall dependability definitions and concepts
- Measuring Site dependability
- Stochastic Models
- Overall Network dependability

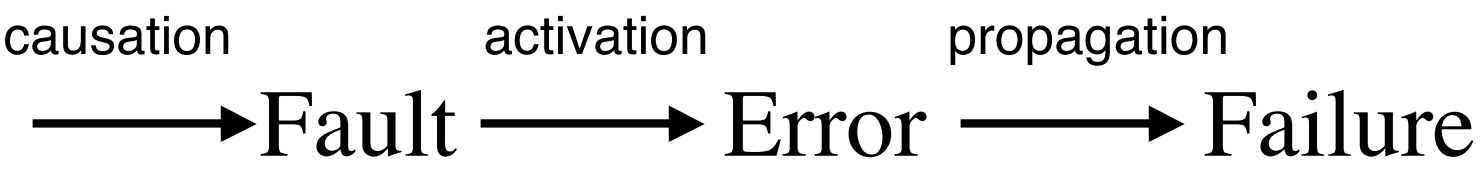
Fundamental Concepts

- Dependable systems must define:
  - What is the service?
    - Observed behavior by users
  - Who/what is the user?
  - What is the service interface?
    - How to user's view the system?
  - What is the function? (intended use)

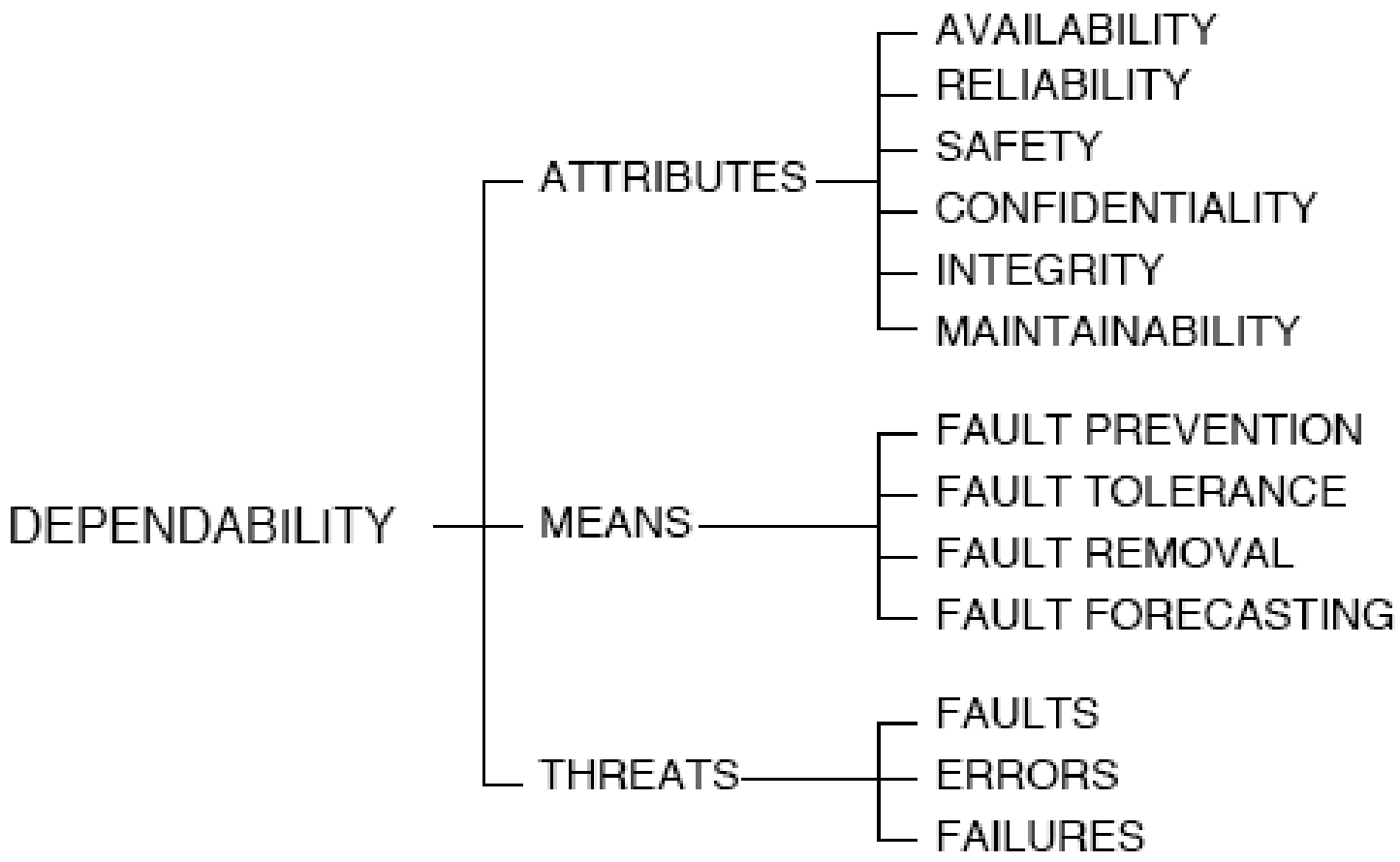
Concepts (2)

- Failure:
  - incorrect service to users
- Outage:
  - time interval of incorrect service
- Error:
  - system state causing failure
- Fault:
  - Cause of error
  - Active (produces error) and latent

Causal Chain



Dependability Definitions



Attributes

- Availability
  - % of time delivering correct service
- Reliability
  - Expected time until incorrect service
- Safety
  - Absence of catastrophic consequences
- Confidentiality
  - Absence of unauthorized disclosure

Attributes (2)

- Integrity
  - Absence of improper states
- Maintainability
  - Ability to undergo repairs
- Security
  - Availability to authorized users
  - Confidentiality
  - Integrity

Means to Dependable Systems

- Fault prevention
- Fault tolerance
- Fault removal
- Fault forecasting

MTTF and MTTR

- Mean Time To Failure (MTTF)
  - Average time to a failure
- Mean Time To Repair
  - Average time under repair
- Availability = % time correct
  = (Time for correct service) / total time
  - Simple model:
    = MTTF/(MTTF+MTTR)

Availability classes and nines

System Type	Unavailability (min/year)	Availability (%)	Class(9's)
Unmanaged	52,560	90%	1
Managed	5,256	99%	2
Well-Managed	526	99.9%	3
Fault-tolerant	53	99.99%	4
High-availability	5	99.999%	5
Very-highly available	0.5	99.9999%	6
Ultra-highly available	0.05	99.99999%	7

Latency

- What about “how long” to perform the service?
- Strict bound:
  - Correct service within T counts, > T = fault
- Statistical bound:
  - N % of requests within <= T

Volume

- Correctness depends on quality of the result
- These systems tend to perform actions on large data sets:
  - Search engines
  - Auctions/pricing
- For a given request, parameterize correctness by % of answers returned as if we used the entire data set

Measuring End-User Availability on the Web: Practical Experience

Matthew Merzbacher

Dan Patterson

UC Berkeley

Introduction

- Availability, performance, QoS important in Web Svcs.
- End user experience -> meaningful benchmark
- Long term experiments attempted to duplicate end user experience
- Find out what the main causes are for downtime as seen by end user.

Driving forces

- Availability/uptime in “9”s not accurate
  - Optimal conditions, not real-world
- Actual uptime to end users include many factors
  - Network, multiple sw layers, client sw/hw
- Need meaningful measure of availability rather one number characterizing unrealistic operating environment

The Experiment

- Undergrads @ Mills College/UC Berkeley devised experiment over several months
- Made hourly contact on a list of several prominent/not-so-prominent sites
- Characterized availability using measures of success, speed, size
- Attempted to pinpoint area of failures

Experiment (cont'd)

- Coded in Java
- Tested local machines as well (to determine baseline and determine local problems
- Random minutes each hour
- Results form 3 types of sites
  - Retailer
  - Search engine
  - Directory service

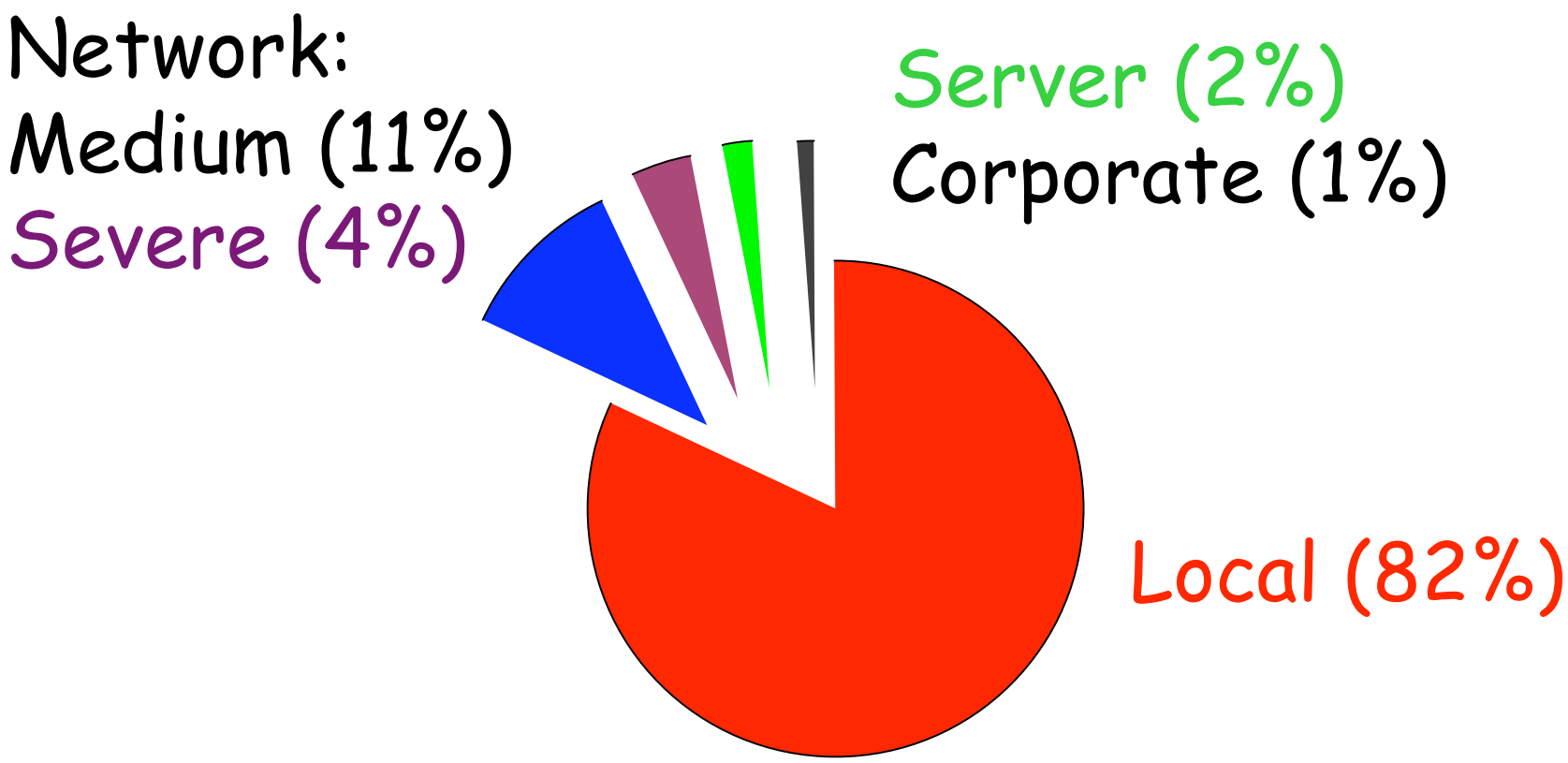
Results

- Availability broken up into sections
  - Raw, local, network, transient
- Kinds of errors broken up into
  - Local, Severe network, Corporate, Medium Network, Server
- Was response upon success partial? How long?

Different Tiers of Availability

	All	Retailer	Search	Directory
Raw (Overall)	**.9305**	**.9311**	**.9355**	**.9267**
Ignoring local problems	**.9888**	**.9887**	**.9935**	**.9857**
Ignoring local and network problems	.9991	.9976	1.00	.9997
Ignoring local, network, and transient problems	.9994	.9984	1.00	.9999

Types of Errors



Local Problems

- Most common problem
- Caused by
  - System crashes, sysadmin problems, config problems, attacks, power outages, etc…
- All had component of human error, but no clear way to solve via preventative measures
- “Local availability dominates the end-user experience”

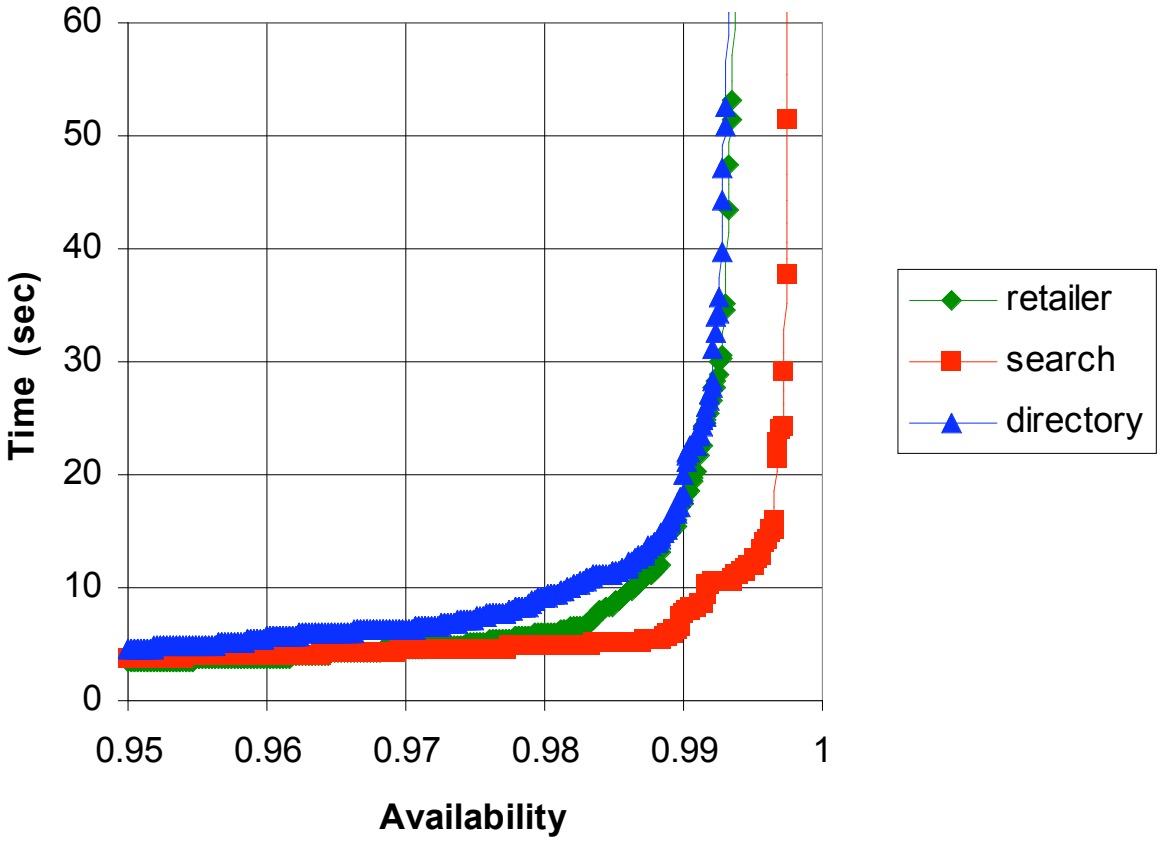
Lost Data and Corporate Failure

- Just because response was received doesn't mean service was available
- Experiment kept track of pages that appeared to be of a drastically different size (smaller) as unavailable (I.e. 404)
- If international versions failed -> corporate failure

Response time

- Wanted to define what “too slow” is
  - Chart availability vs. time
  - Asymptotic towards availability of 1
  - Choose threshold, all response times > considered unavailable
  - Client errors most frequent type of error, then transient network

How long should we wait?



Retrying

- To users, unavailability leads to retry at least once
- How effective is a retry?
  - Need to test for persistence of failures
  - Consistent failures indicate fault @/near server
- Persistent, non-local failures
  - Domain dependent

Retrying (cont'd)

- Retry period of 1 hour unrealistic
- As in brick & mortar, clients have choice
- #retries, time btwn retries, etc based on domain/user dependent factors
  - Uniqueness, import, loyalty, transience

Effect of retry

Error Type	All	Retailer	Search	Directory
Client	**0.267**	**0.271**	**0.265**	**0.265**
Medium Network	**0.862**	**0.870**	**0.929**	**0.838**
Severe Network	**0.789**	**0.923**	**1.00**	**0.689**
Server	**0.911**	**0.786**	**1.00**	**0.96**
Corporate	**0.421**	**0.312**	**1.00**	**n/a**

Green > 80%

Red < 50%

Conclusion

- Successful in modeling user experience
- 93% Raw, 99.9% removing local/short-term errors
- Retry produced better availability, reduced error 27% in local, 83% non-local
- Factoring in retries produces 3 “9s” of availability.
- Retry doesn’t help for local errors
  - User may be aware of the problem and therefore less frustrated by it

Future Work

- Continue experiment, refine availability stats
- Distribute experiment across distant sites to analyze source of errors
- Better experiments to determine better the effects of retry
- With the above, we can pinpoint source of failures and make more reliable systems.

Stochastic Analysis of Computer Networks

- Capture probabilistic behavior as a function of time

- Formalisms:
  - Markov Chains
    - Discrete
    - Continuous
  - Petri Nets

Markov Chains

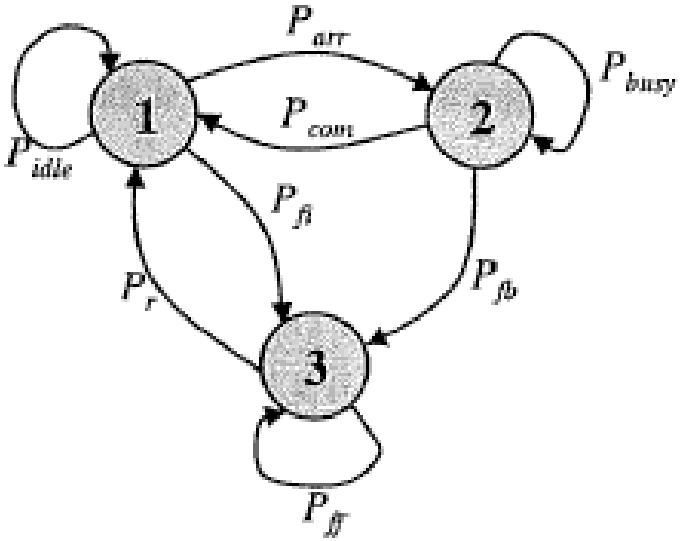
- States
- Transitions
- Transition probabilities
- Evolution over time
- Compute:
  - Average time spent in a state
    - Fraction of time time
  - Expected time to reach a state
  - “Reward” for each state

Discrete Markov chains

- States of the system
- Time modeled in discrete, uniform steps
  - (e.g. every minute)
- Each state has a set of transition probabilities to other states for each time step
  - Sum of probabilities == 1

Discrete Markov Chain

"Simple Computer" Example



Graphical representation

$X = 1$ computer idle
$X = 2$ computer working
$X = 3$ computer failed

$$P = \begin{bmatrix} P_{idle} & P_{arr} & P_{fi} \\ P_{com} & P_{busy} & P_{fb} \\ P_r & 0 & P_{ff} \end{bmatrix}$$

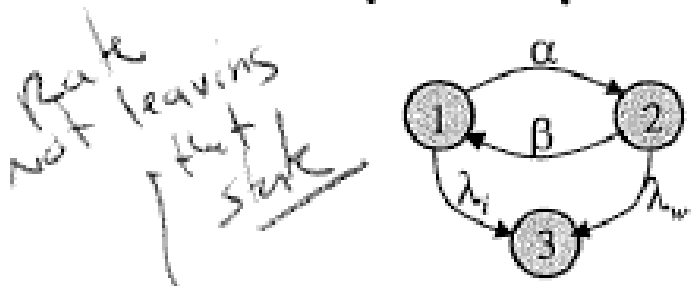
Probability transition matrix representation

Continuous Time Markov Chains

- States of the system (as before)
- Transitions are rates with exponential distributions
  - Some event arrives at rate λ with an an exponential interarrival time

Continuous Time Markov Chain

"Simple Computer" CTMC

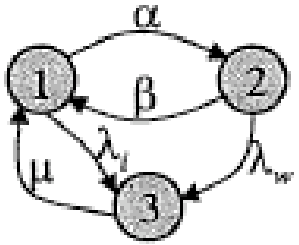


Let $X = 1$ represent "the system is idle," $X = 2$ "the system is working," and $X = 3$ a failure.

$$Q = \begin{bmatrix} -(\alpha+\lambda_i) & \alpha & \lambda_i \\ \beta & -(\beta+\lambda_w) & \lambda_w \\ 0 & 0 & 0 \end{bmatrix}$$

Rows sum to zero for steady state behavior

If the computer is repaired with rate μ, the new CTMC looks like



$$Q = \begin{bmatrix} -(\alpha+\lambda_i) & \alpha & \lambda_i \\ \beta & -(\beta+\lambda_w) & \lambda_w \\ \mu & 0 & -\mu \end{bmatrix}$$

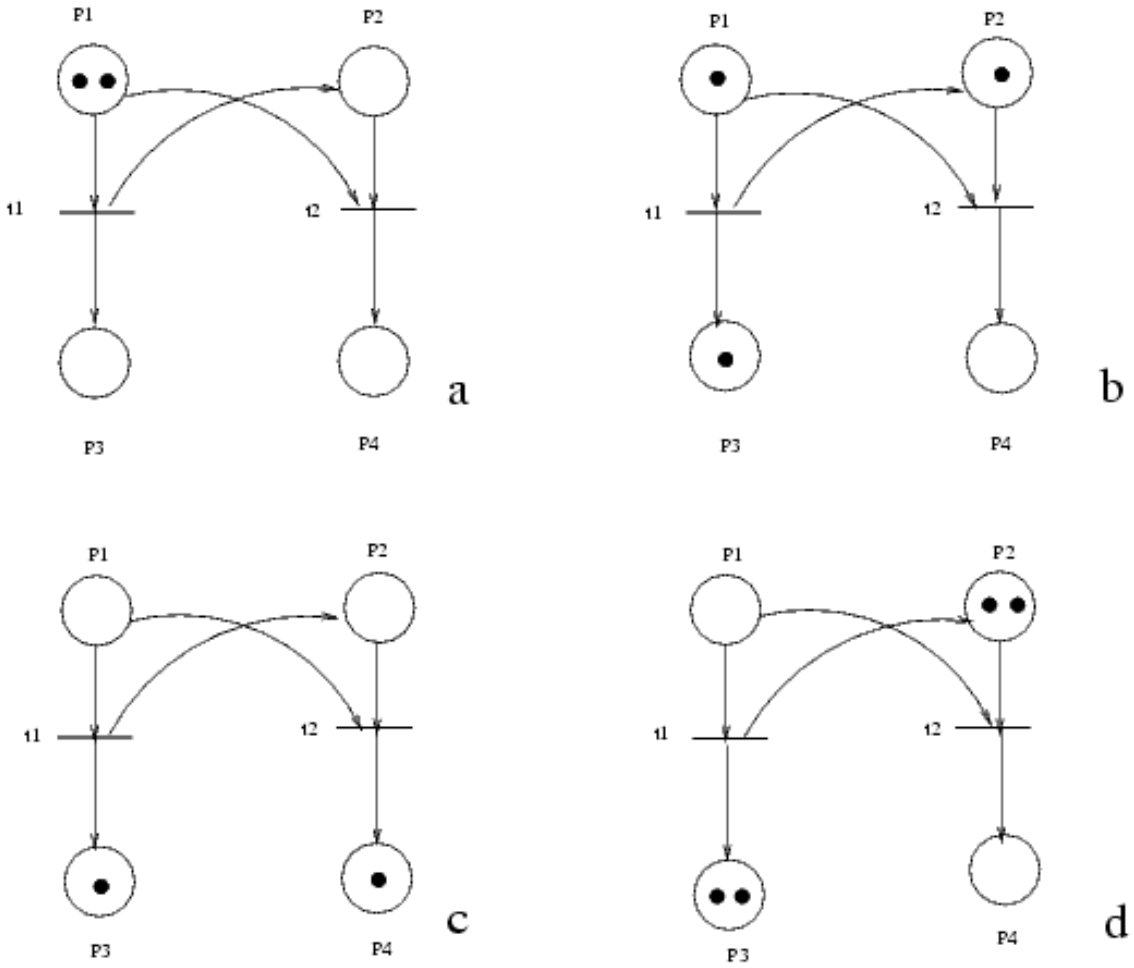
Answers from the CTMC model

- Availability at time t?
- Steady state availability?
- Expected time to failure?
- Expected number of jobs lost due to failure over [0,t]?
- Expected number of jobs served before failure?
- Expected throughput given failures and repairs

Petri Nets

- Higher level formalism
- Components:
  - Places
  - Transitions
  - Arcs
  - Weights
  - Markings

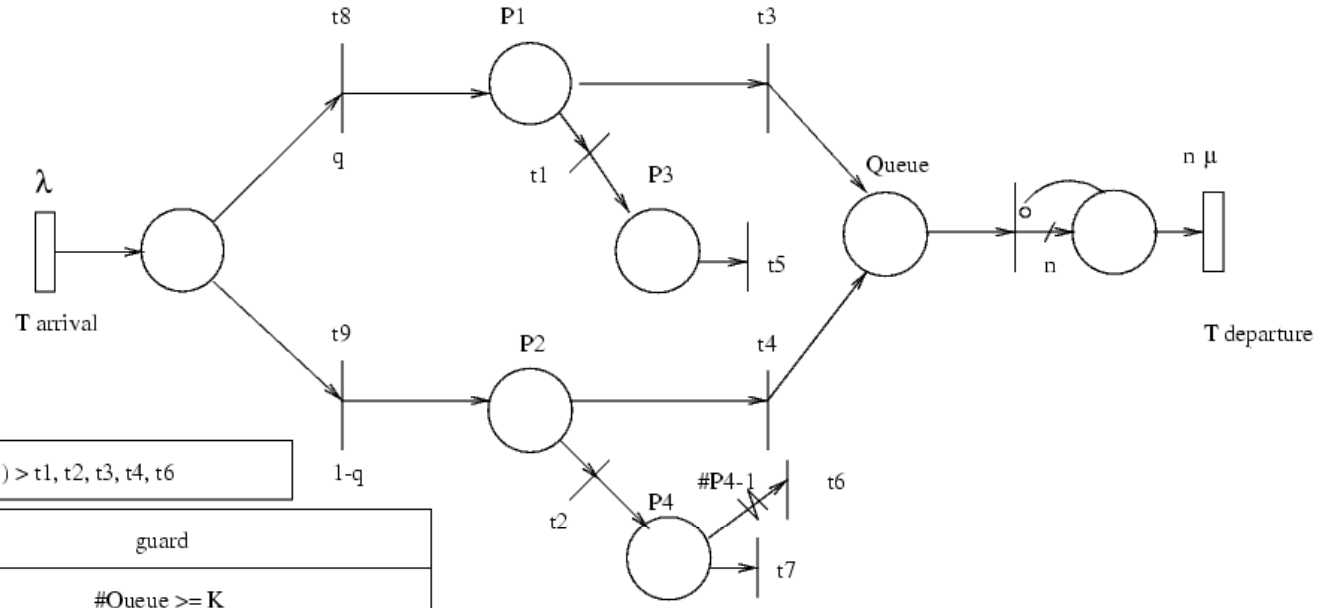
Petri Nets


P1
P2
t1
t2
P3
P4
a
P1
P2
t1
t2
P3
P4
b
P1
P2
t1
t2
P3
P4
c
P1
P2
t1
t2
P3
P4
d

General Stochastic PN

- Either exponential or instant departures

ATM Example



priority : (t5, t7) > t1, t2, t3, t4, t6

transition	guard
t1	#Queue >= K
t2	#Queue = N or #P3 > 0 or #P4 > 0
t3	#Queue < K
t4	#Queue < N and #P3 = #P4 = 0
t5	#P1 > 0
t6	#P4 > 1
t7	#P1 > 0

guards for no-control syetem

transition	guard
t1	#Queue =N
t2	#Queue = N
t3	#Queue < N
t4	#Queue < N