Multiple Class G-Networks with Restart

Jean-Michel Fourneau, **Katinka Wolter**, Philipp Reinecke,
Tilman Krauß and Alexandra Danilkina

PRiSM, CNRS
Université de Versailles
France

Freie Universität Berlin
Institut für Informatik

HP Labs Bristol
Long Down Avenue,
Bristol, UK

ISCAS, 26 September 2013

Table of Contents

1 The restart method

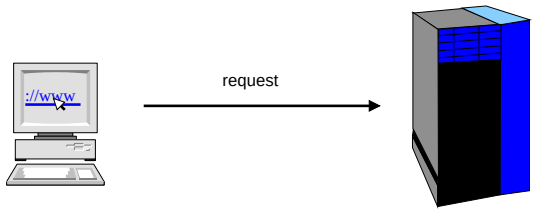
2 Evaluation Approaches

3 The model

4 Examples

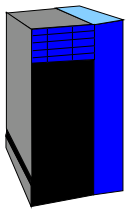
5 Conclusions

The restart method



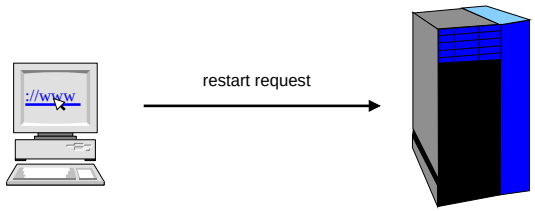
- Restart: A client sends a request.

The restart method



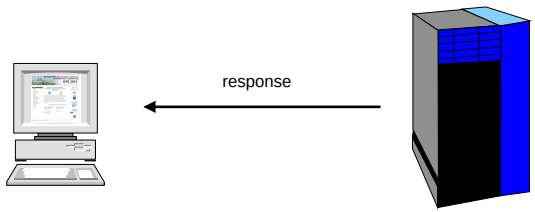
- Restart: A client sends a request. If there is no response within a reasonable time,

The restart method



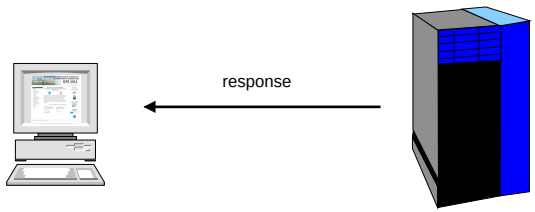
- Restart: A client sends a request. If there is no response within a reasonable time, the request is repeated

The restart method



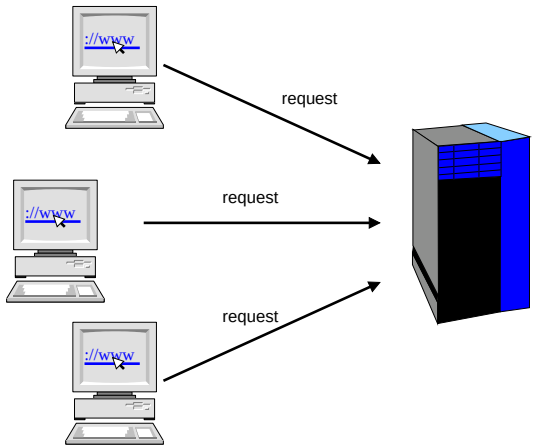
- Restart: A client sends a request. If there is no response within a reasonable time, the request is repeated

The restart method



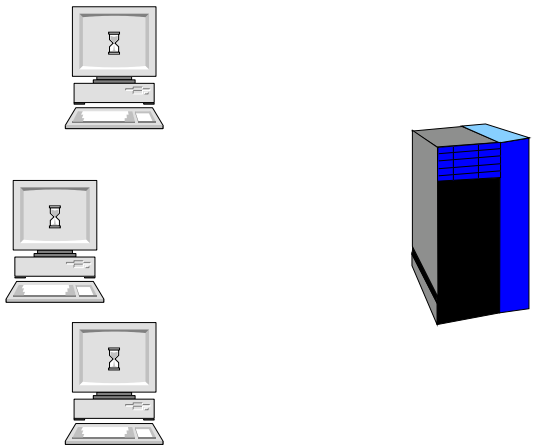
- Restart: A client sends a request. If there is no response within a reasonable time, the request is repeated
- Restart may reduce response-times

The restart method



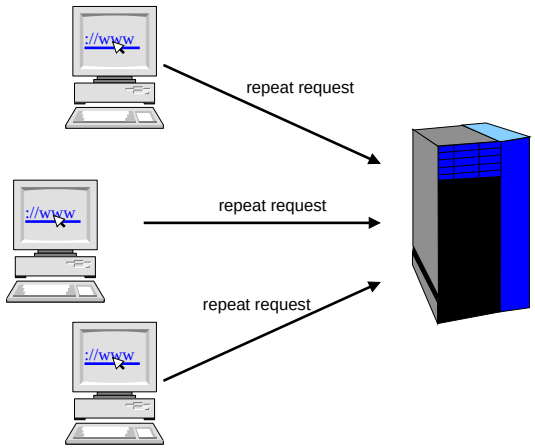
- Restart: A client sends a request. If there is no response within a reasonable time, the request is repeated
- Restart may reduce response-times
- Many clients send a request.

The restart method



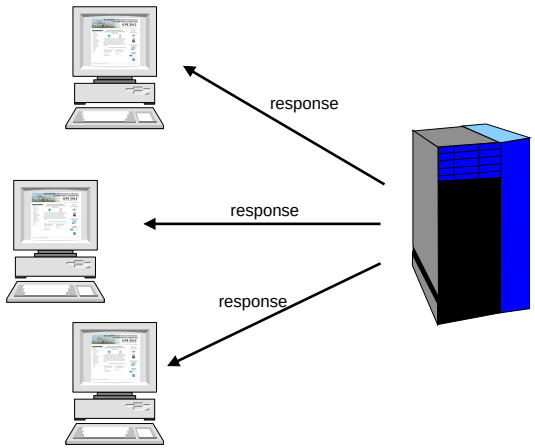
- Restart: A client sends a request. If there is no response within a reasonable time, the request is repeated
- Restart may reduce response-times
- Many clients send a request. If there is no response within a reasonable time,

The restart method



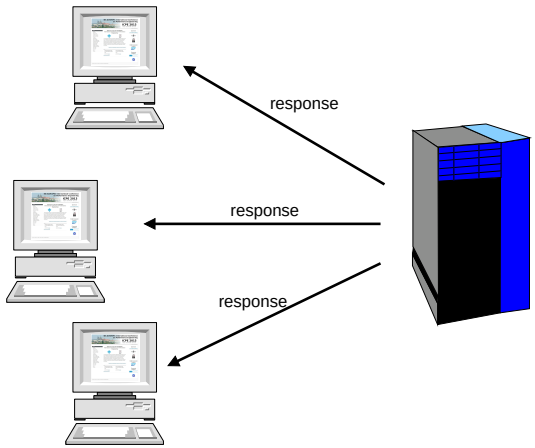
- Restart: A client sends a request. If there is no response within a reasonable time, the request is repeated
- Restart may reduce response-times
- Many clients send a request. If there is no response within a reasonable time, the request is repeated

The restart method



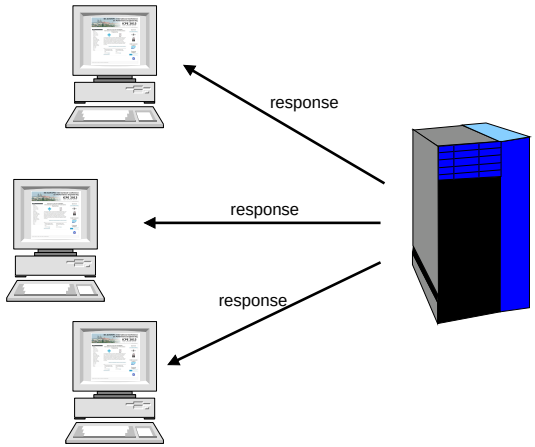
- Restart: A client sends a request. If there is no response within a reasonable time, the request is repeated
- Restart may reduce response-times
- Many clients send a request. If there is no response within a reasonable time, the request is repeated
- Question: Will restart reduce response-times for all?

The restart method



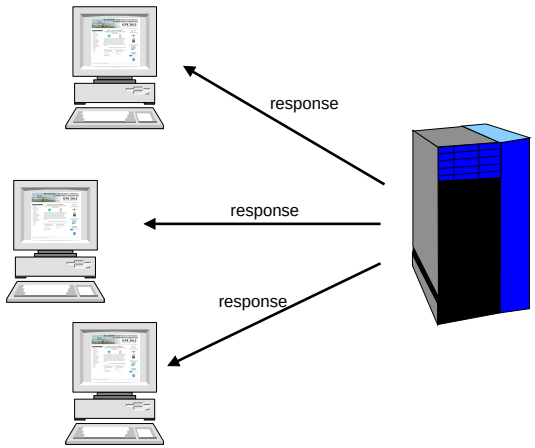
- Restart: A client sends a request. If there is no response within a reasonable time, the request is repeated
- Restart may reduce response-times
- Many clients send a request. If there is no response within a reasonable time, the request is repeated
- Question: Will restart reduce response-times for all?
- Insight useful for

The restart method



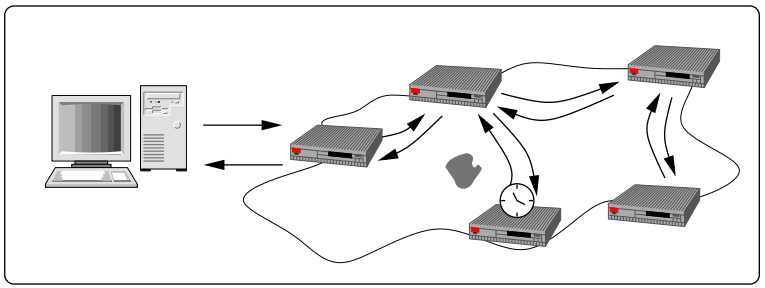
- Restart: A client sends a request. If there is no response within a reasonable time, the request is repeated
- Restart may reduce response-times
- Many clients send a request. If there is no response within a reasonable time, the request is repeated
- Question: Will restart reduce response-times for all?
- Insight useful for
  - Optimising software performance

The restart method



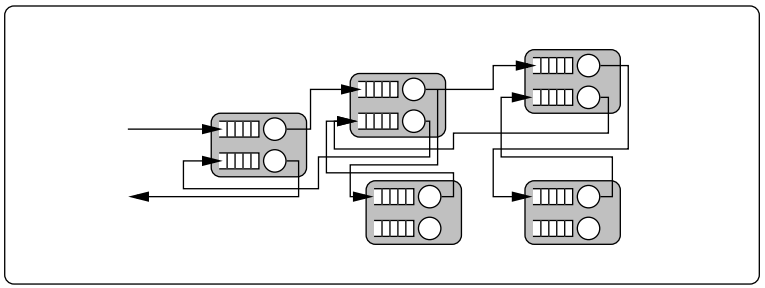
- Restart: A client sends a request. If there is no response within a reasonable time, the request is repeated
- Restart may reduce response-times
- Many clients send a request. If there is no response within a reasonable time, the request is repeated
- Question: Will restart reduce response-times for all?
- Insight useful for
  - Optimising software performance
  - Development of benchmarks

Evaluation Approaches



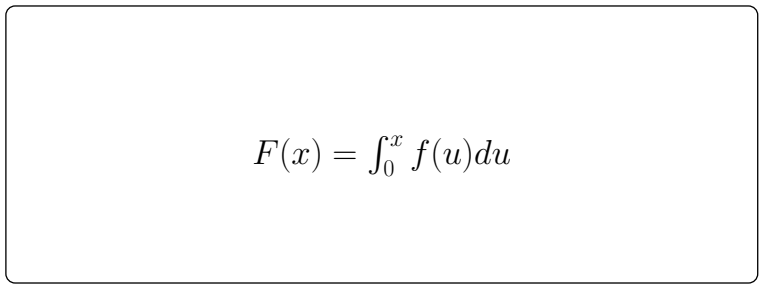
- Experimentation on test-beds – low abstraction level
  - Set up the system in the lab
  - Cost and time constraints
  - Results do not generalise

Evaluation Approaches



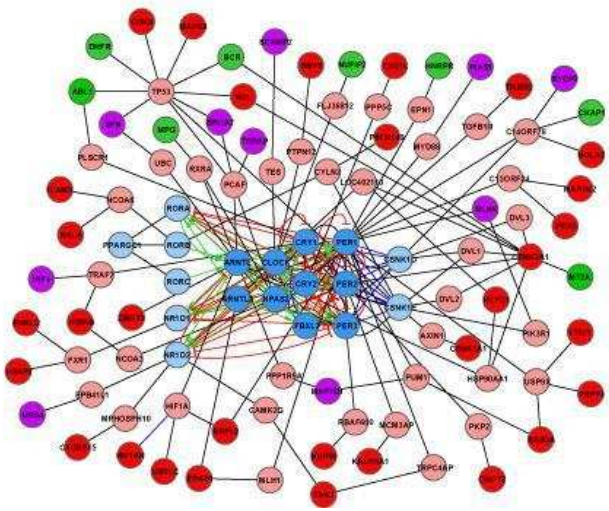
- Experimentation on test-beds – low abstraction level
  - Set up the system in the lab
  - Cost and time constraints
  - Results do not generalise
- Simulations – medium abstraction level
  - Build a simulation (e.g. NS-2, OMNeT++, Möbius)
  - Give slightly more general results than measurement studies
  - Results are less realistic

Evaluation Approaches



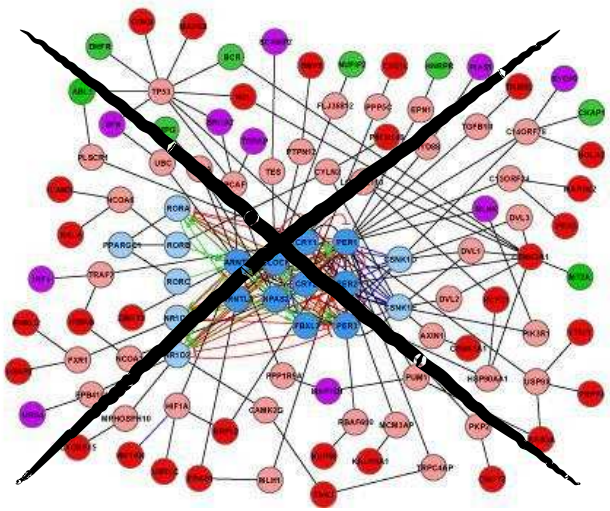
- Experimentation on test-beds – low abstraction level
  - Set up the system in the lab
  - Cost and time constraints
  - Results do not generalise
- Simulations – medium abstraction level
  - Build a simulation (e.g. NS-2, OMNeT++, Möbius)
  - Give slightly more general results than measurement studies
  - Results are less realistic
- Analytical Approaches – high abstraction level
  - Formalise problem
  - Give general insights
  - Results might be far from reality

G-networks



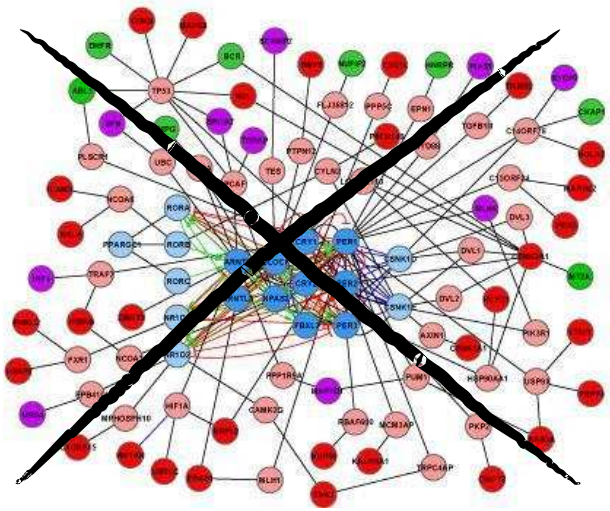
- one could guess, a G-network is a genetic network

G-networks



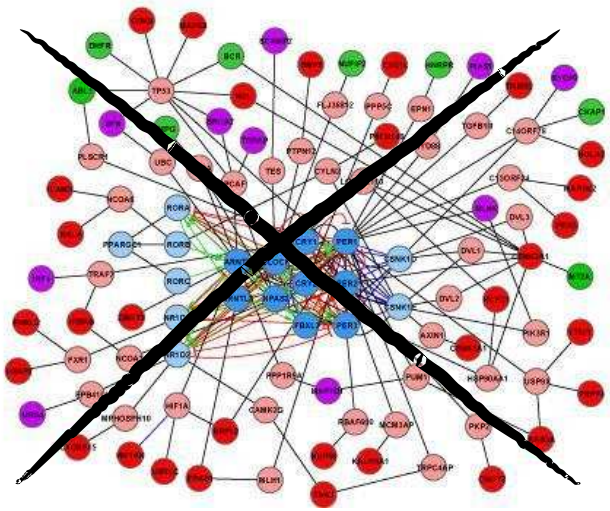
- one could guess, a G-network is a genetic network
- it is not.

G-networks



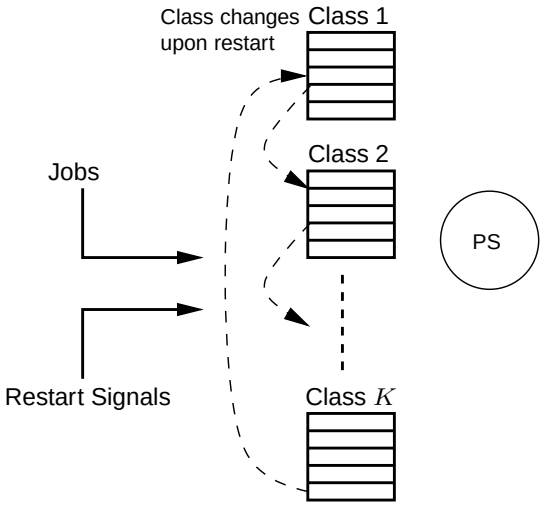
- one could guess, a G-network is a genetic network
- it is not.
- G as generalized queueing network or Gelenbe network

G-networks



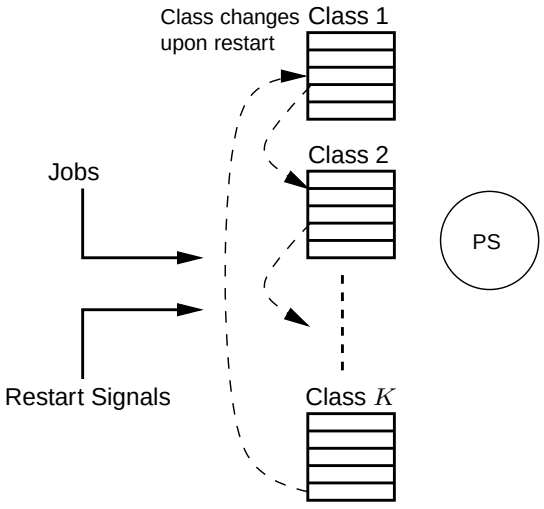
- one could guess, a G-network is a genetic network
- it is not.
- G as generalized queueing network or Gelenbe network
- A G-network is an open queueing network with several types of customers
  - regular jobs
  - negative customers, signals
  - (signals between queues)

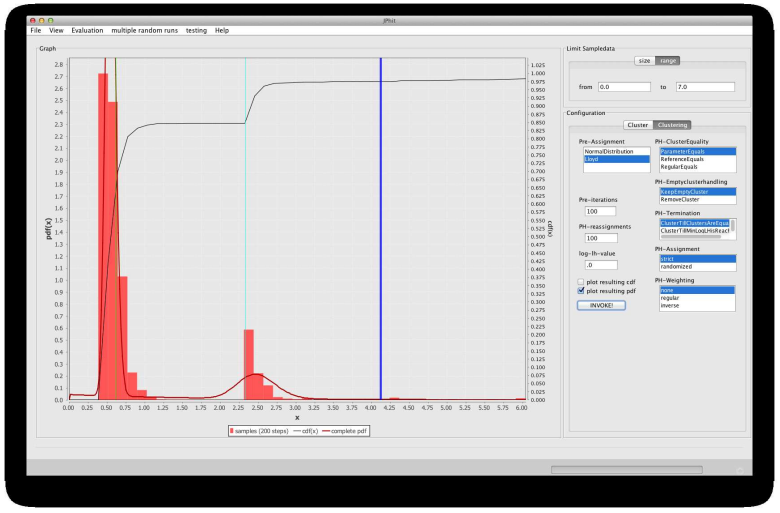
G-networks



- Jobs arrive into one of K classes

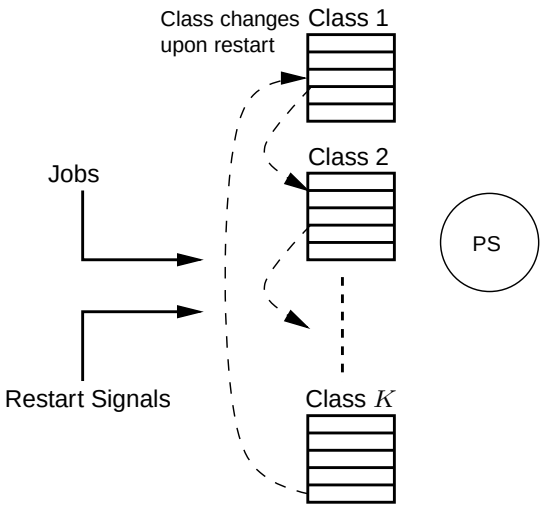
G-networks

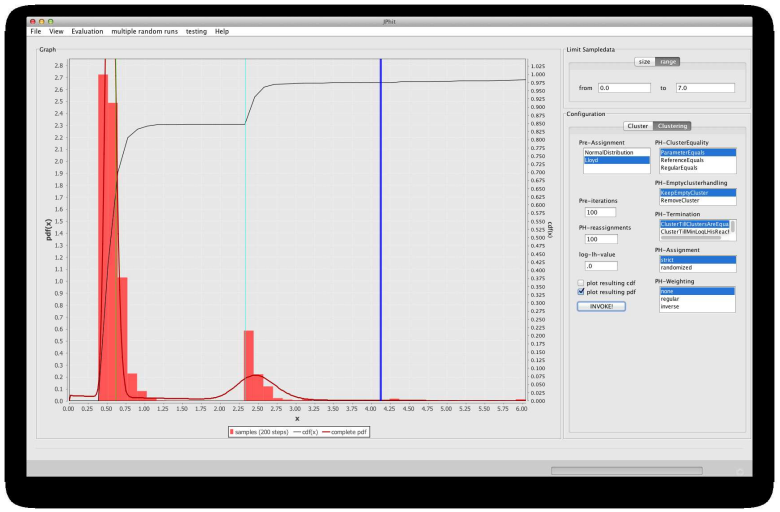




- Jobs arrive into one of K classes
- Service time PH-distributed

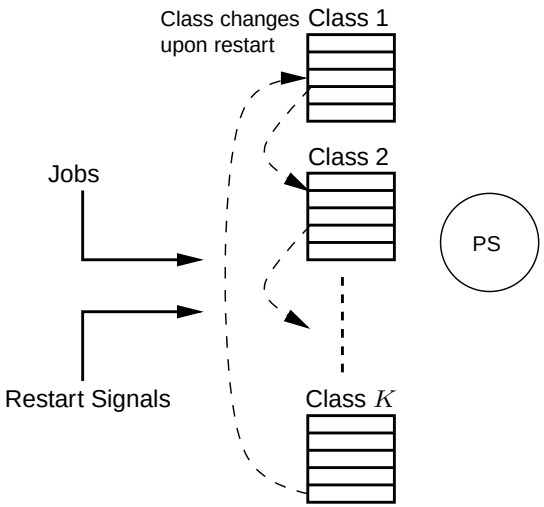
G-networks

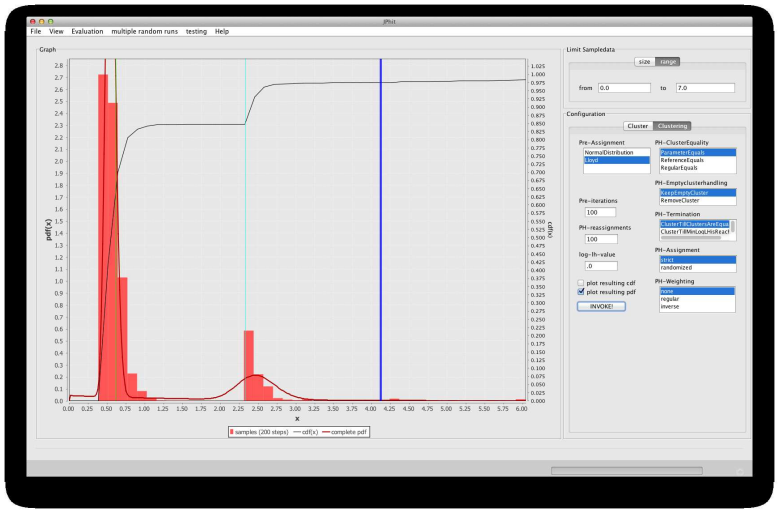




- Jobs arrive into one of K classes
- Service time PH-distributed
- One class of signals (restarts) with success probability

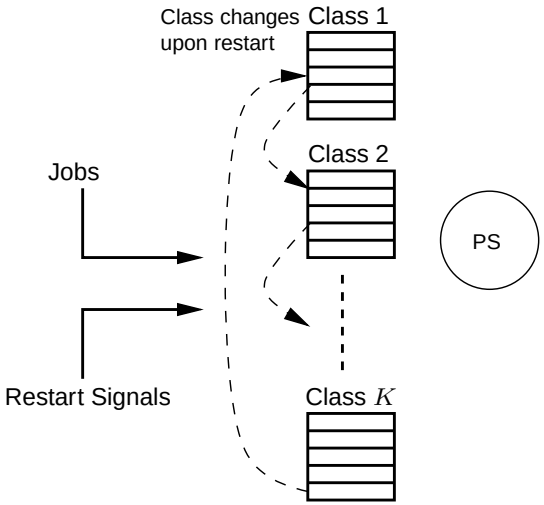
G-networks





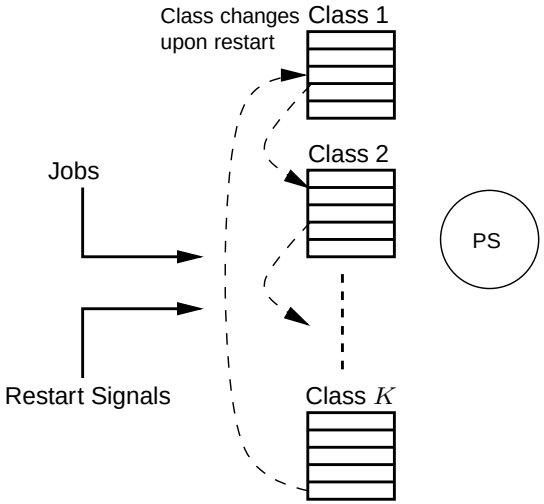
- Jobs arrive into one of K classes
- Service time PH-distributed
- One class of signals (restarts) with success probability
- Upon restart a job changes class, routing probability, success probabilities

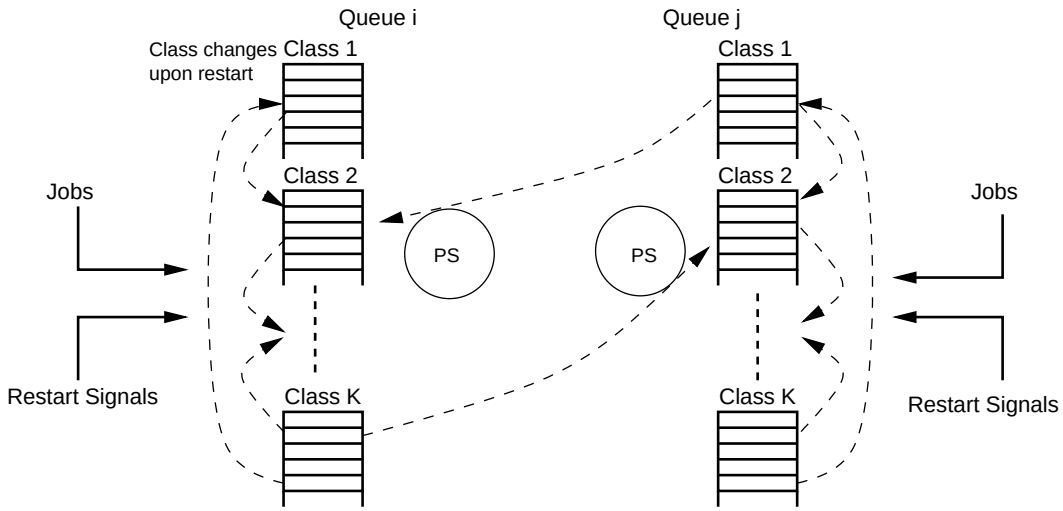
G-networks



- Jobs arrive into one of K classes
- Service time PH-distributed
- One class of signals (restarts) with success probability
- Upon restart a job changes class, routing probability, success probabilities
- Markovian model

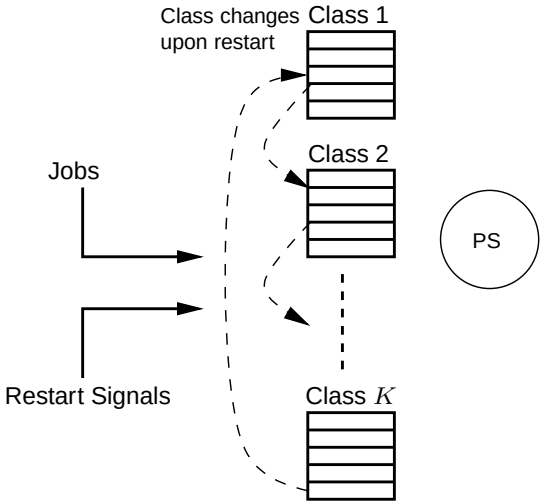
G-networks

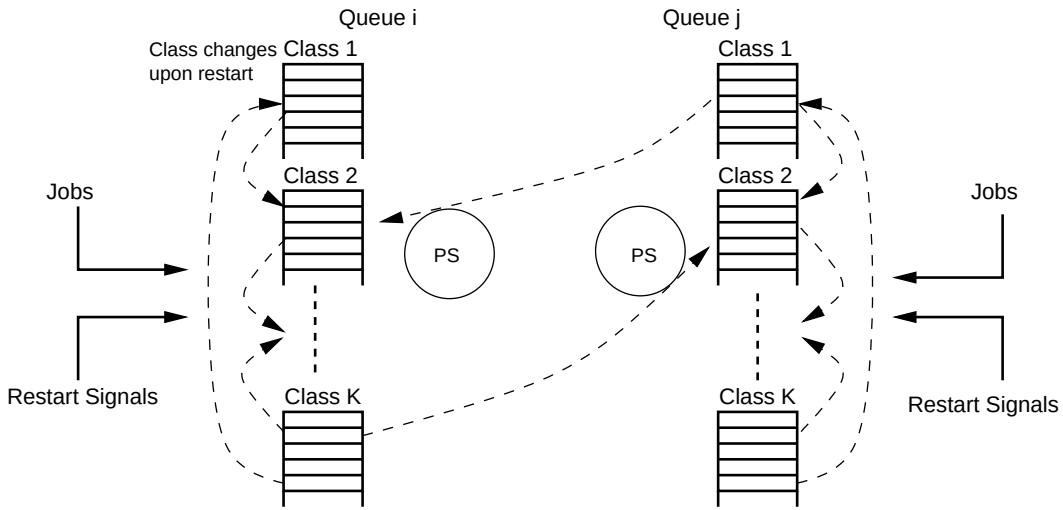




- Jobs arrive into one of K classes
- Service time PH-distributed
- One class of signals (restarts) with success probability
- Upon restart a job changes class, routing probability, success probabilities
- Markovian model
- Product-form solution

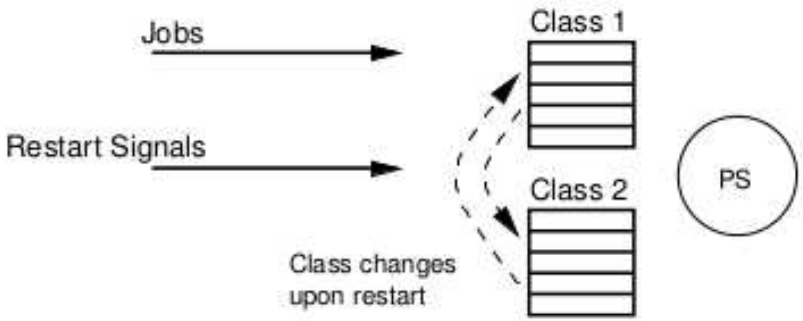
G-networks





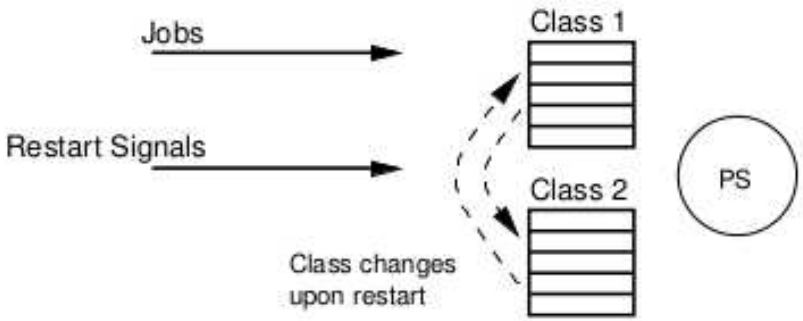
- Jobs arrive into one of K classes
- Service time PH-distributed
- One class of signals (restarts) with success probability
- Upon restart a job changes class, routing probability, success probabilities
- Markovian model
- Product-form solution
- Derivation of standard queueing metrics straight forward

First example: iid services



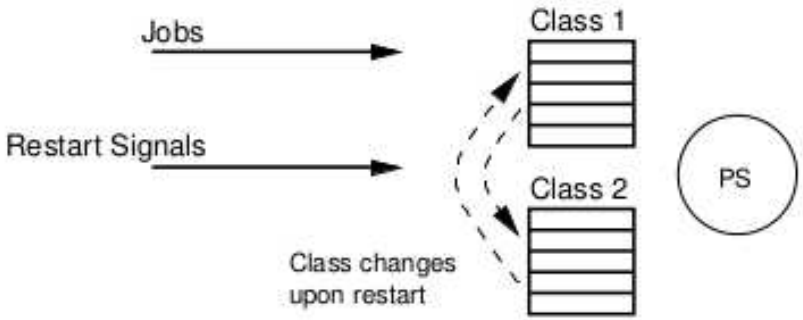
- Model represents infinite restarts

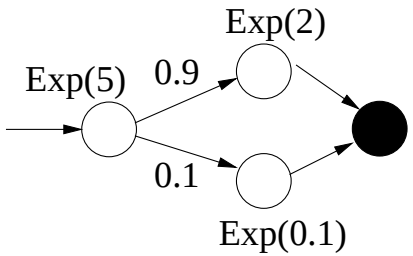
First example: iid services



- Model represents infinite restarts
- Jobs arrive to class 1 at rate 0.012

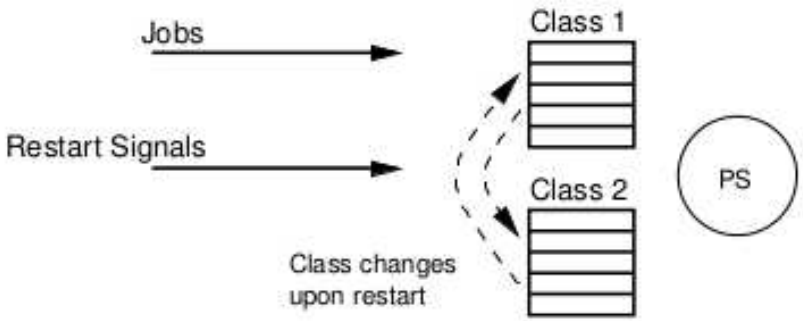
First example: iid services

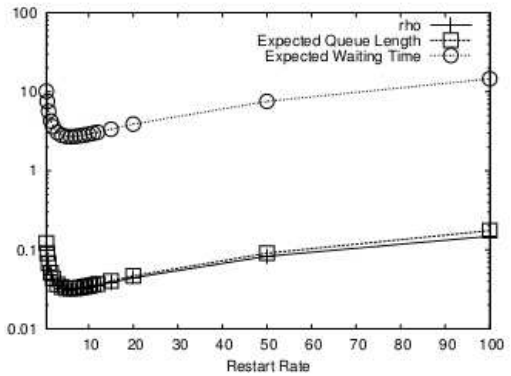




- Model represents infinite restarts
- Jobs arrive to class 1 at rate 0.012
- PH-distributed service time, $scv = 6.7539$

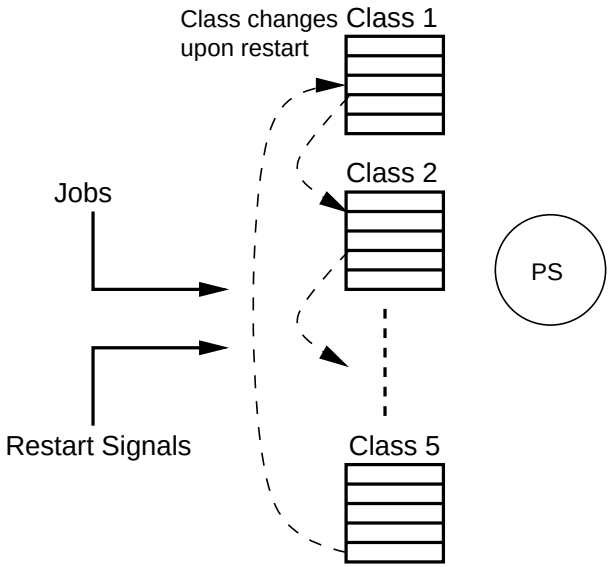
First example: iid services





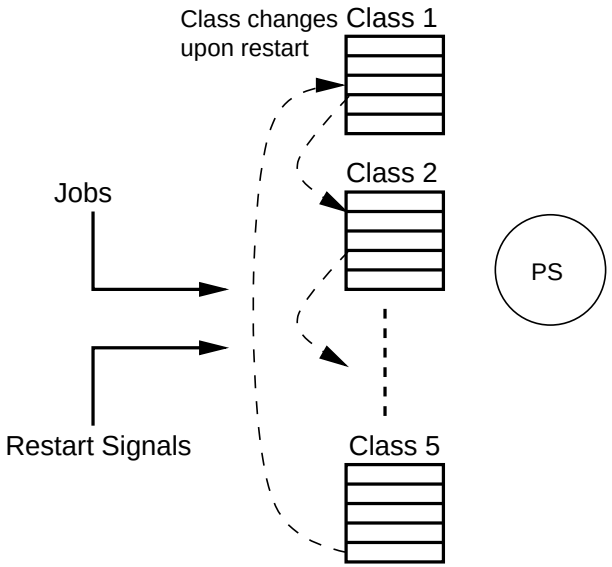
- Model represents infinite restarts
- Jobs arrive to class 1 at rate 0.012
- PH-distributed service time, $scv = 6.7539$
- Determine utilisation, expected queue length and expected waiting time

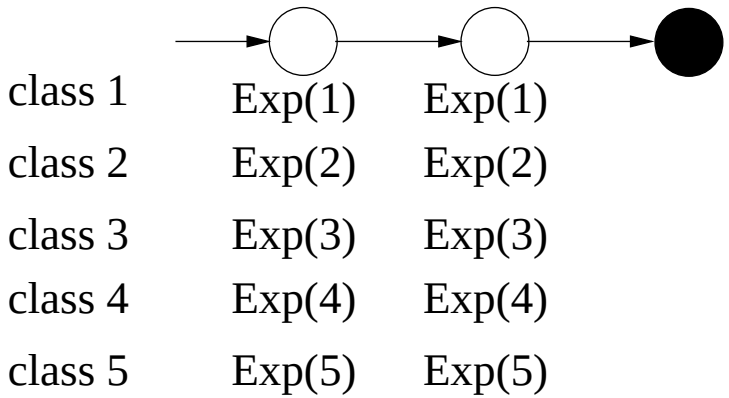
Second example: 5 classes



- Jobs arrive only to class 1 at rate 0.1

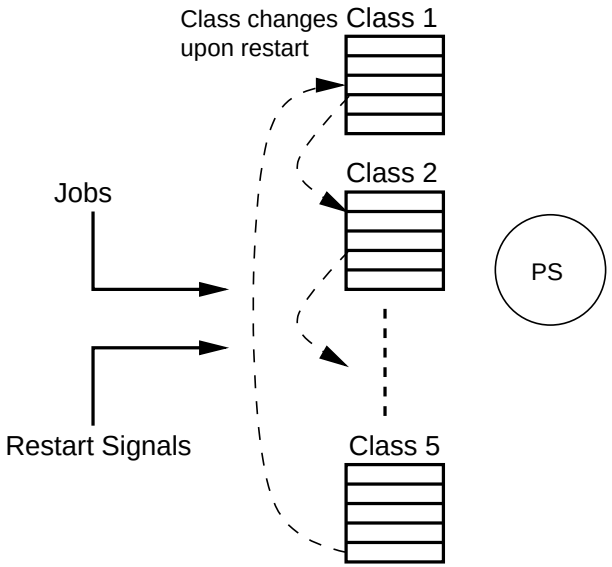
Second example: 5 classes

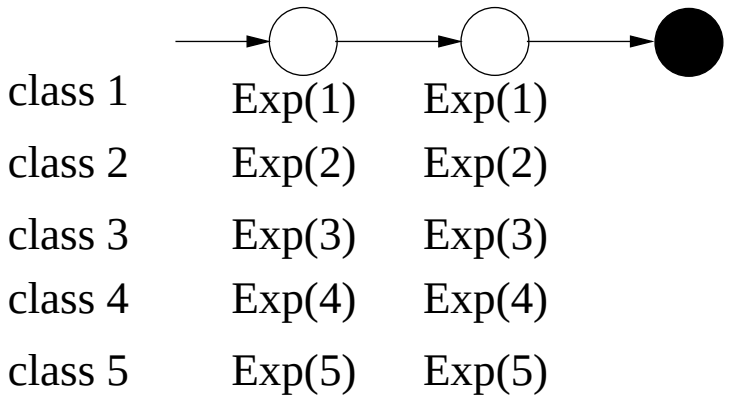




- Jobs arrive only to class 1 at rate 0.1
- Successively faster Erlang(2) service, i.e. $\mu^{(k,p)} = (1, 2, 3, 4, 5)$

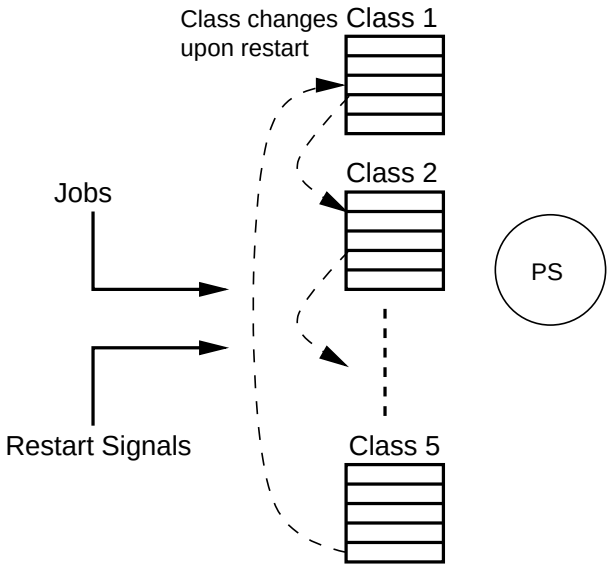
Second example: 5 classes





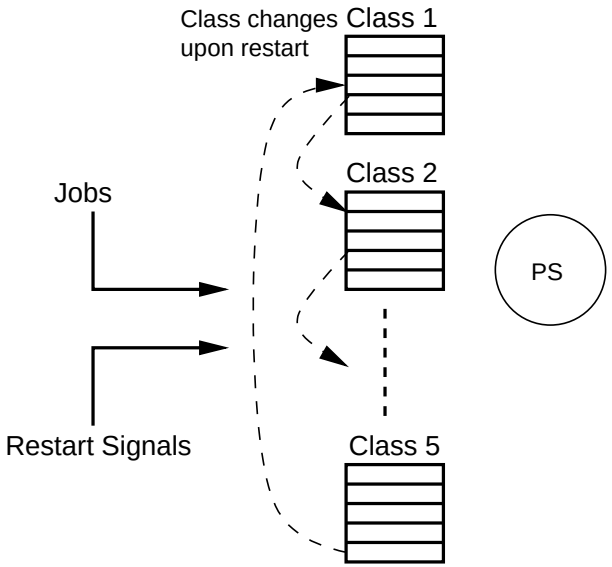
- Jobs arrive only to class 1 at rate 0.1
- Successively faster Erlang(2) service, i.e. $\mu^{(k,p)} = (1, 2, 3, 4, 5)$
- Restart with resume semantics - work is not lost

Second example: 5 classes



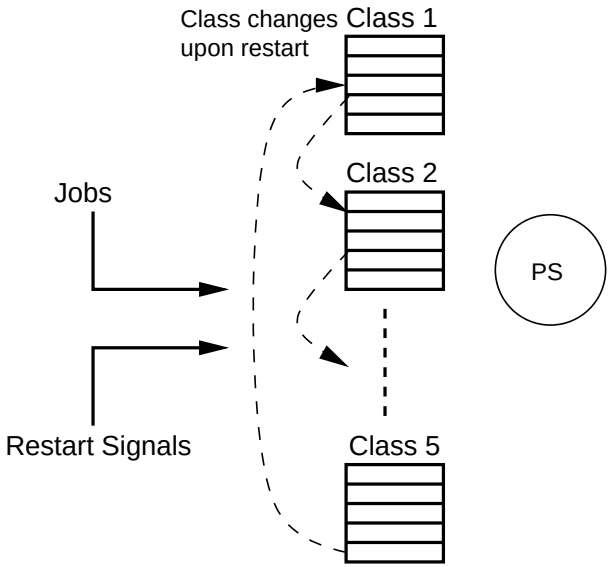
- Jobs arrive only to class 1 at rate 0.1
- Successively faster Erlang(2) service, i.e. $\mu^{(k,p)} = (1, 2, 3, 4, 5)$
- Restart with resume semantics - work is not lost
- Completed jobs leave with prob 1.

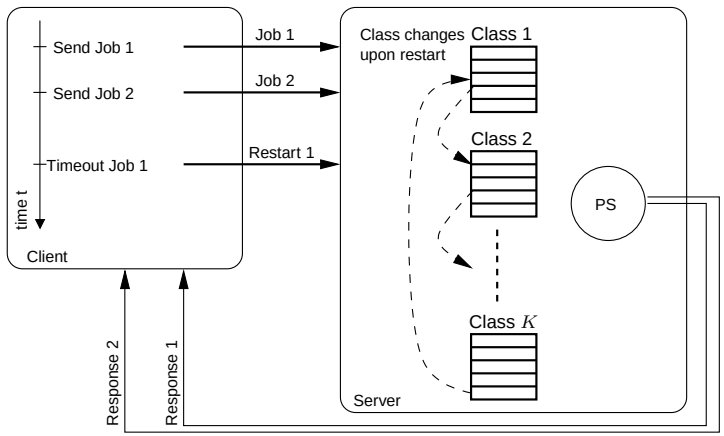
Second example: 5 classes



- Jobs arrive only to class 1 at rate 0.1
- Successively faster Erlang(2) service, i.e. $\mu^{(k,p)} = (1, 2, 3, 4, 5)$
- Restart with resume semantics - work is not lost
- Completed jobs leave with prob 1.
- Restart in class 5 starts job anew

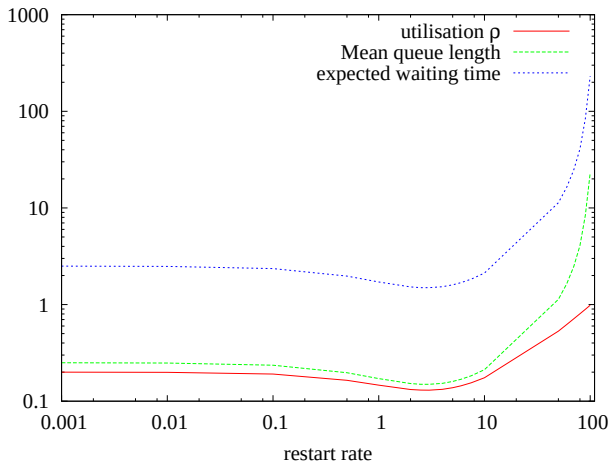
Second example: 5 classes

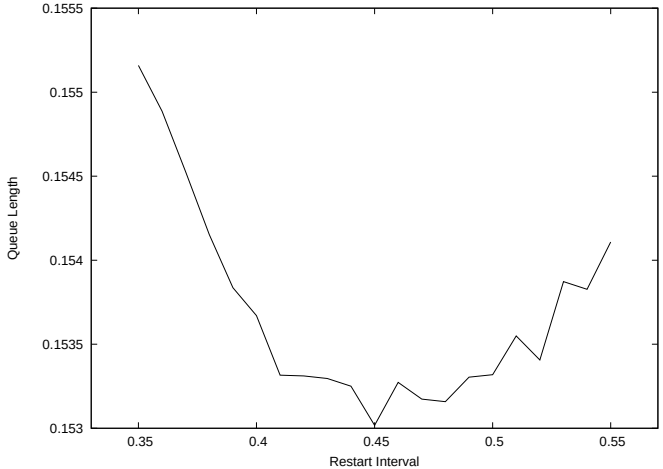




- Jobs arrive only to class 1 at rate 0.1
- Successively faster Erlang(2) service, i.e. $\mu^{(k,p)} = (1, 2, 3, 4, 5)$
- Restart with resume semantics - work is not lost
- Completed jobs leave with prob 1.
- Restart in class 5 starts job anew
- Compare results with simulation using SFERA

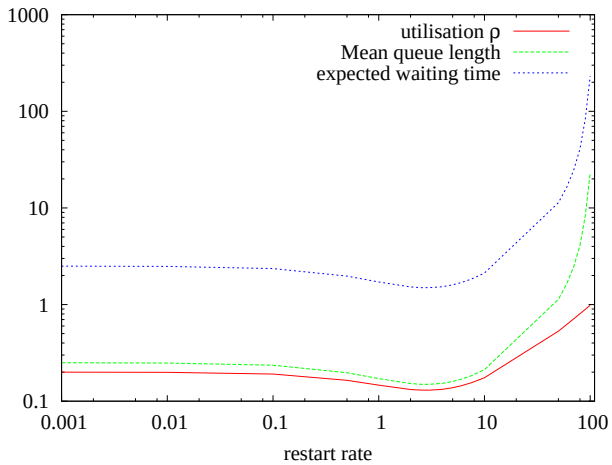
Results

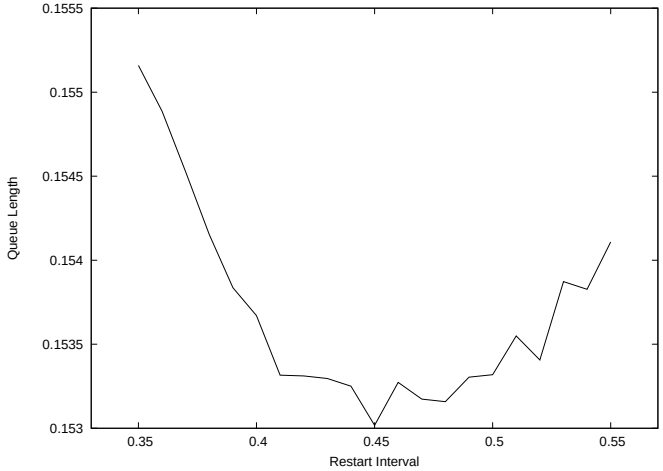




- Expected queue length and expected waiting time for different values of the restart rate
- Average queue length and average response time for the simulation

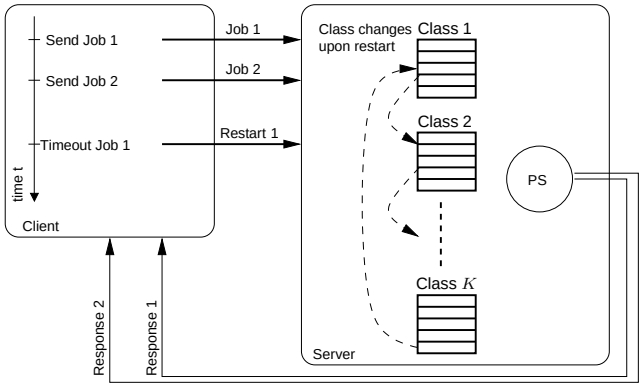
Results





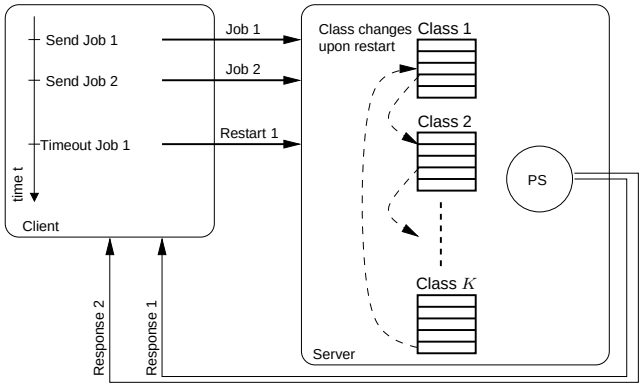
- Expected queue length and expected waiting time for different values of the restart rate
- Average queue length and average response time for the simulation
- Both have minimum for similar restart rate
- Why are simulation and G-network not exactly the same?

What is the difference between SFERA and G-network



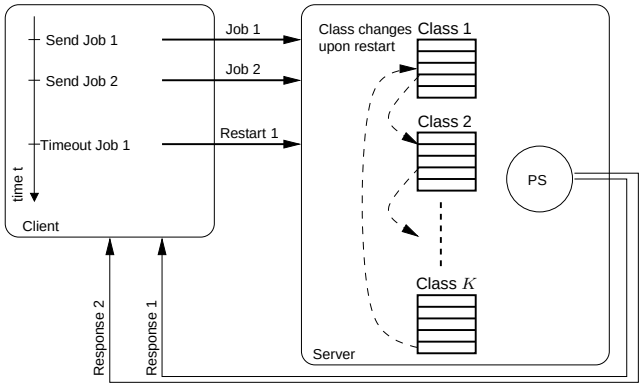
- Simulation assigns a timeout to each job

What is the difference between SFERA and G-network



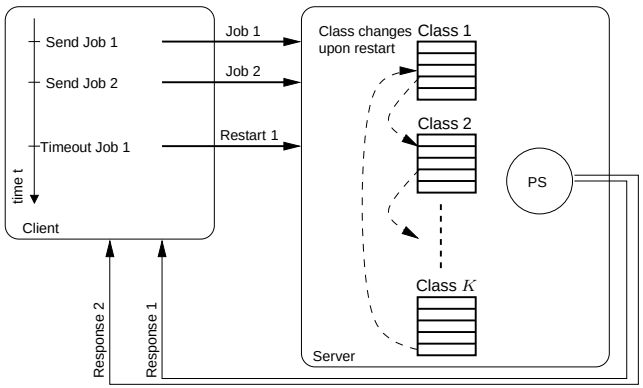
- Simulation assigns a timeout to each job
- PS strategy means processing speed depends on queue length

What is the difference between SFERA and G-network



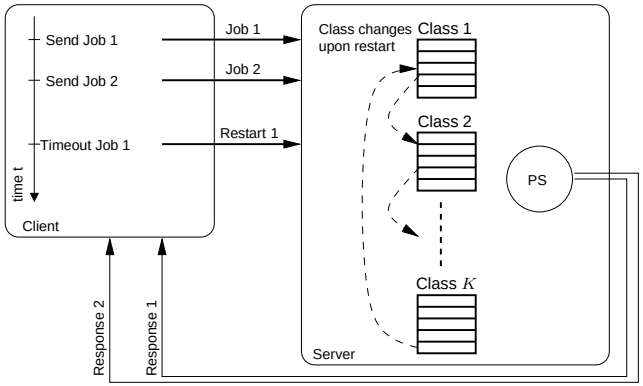
- Simulation assigns a timeout to each job
- PS strategy means processing speed depends on queue length
- Response time depends on queue length

What is the difference between SFERA and G-network



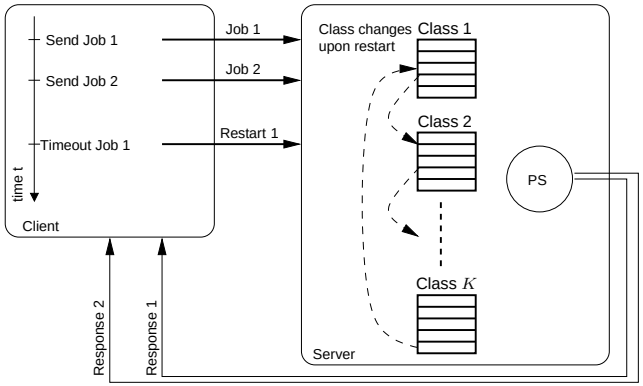
- Simulation assigns a timeout to each job
- PS strategy means processing speed depends on queue length
- Response time depends on queue length
- Each event (arrival, departure) results in reschedule of all future events

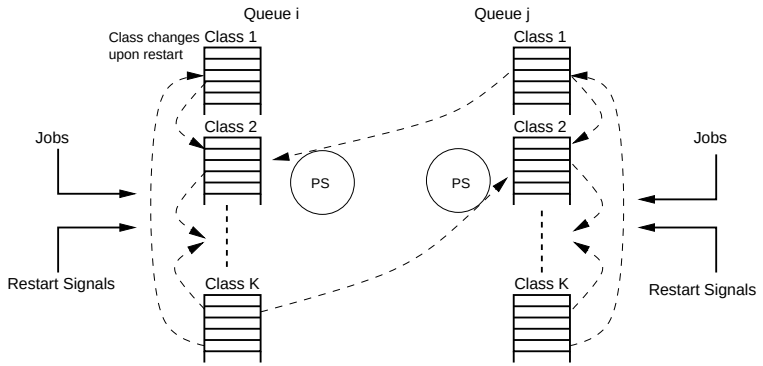
What is the difference between SFERA and G-network



- Simulation assigns a timeout to each job
- PS strategy means processing speed depends on queue length
- Response time depends on queue length
- Each event (arrival, departure) results in reschedule of all future events
- Timeout always remains attached to a job

What is the difference between SFERA and G-network

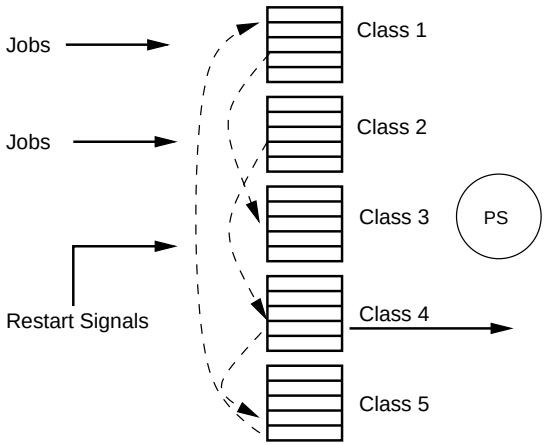




- Simulation assigns a timeout to each job
- PS strategy means processing speed depends on queue length
- Response time depends on queue length
- Each event (arrival, departure) results in reschedule of all future events
- Timeout always remains attached to a job

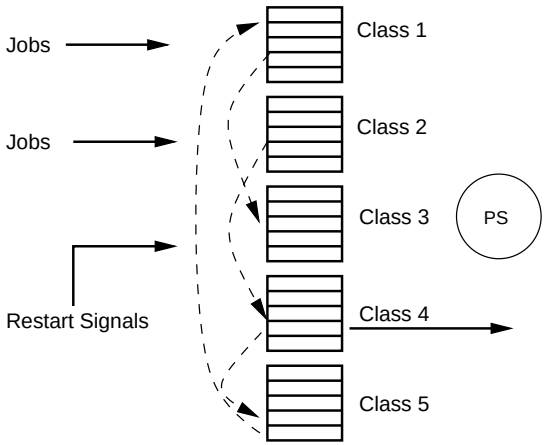
- G-network restarts random job
- Jobs have no 'age'

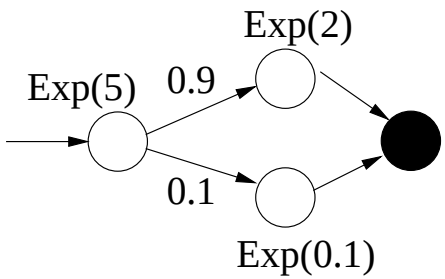
Third example: more complex service times



- Job arrivals to class 1 ($\lambda_1 = 0.004$) and class 2 ($\lambda_2 = 0.01$)

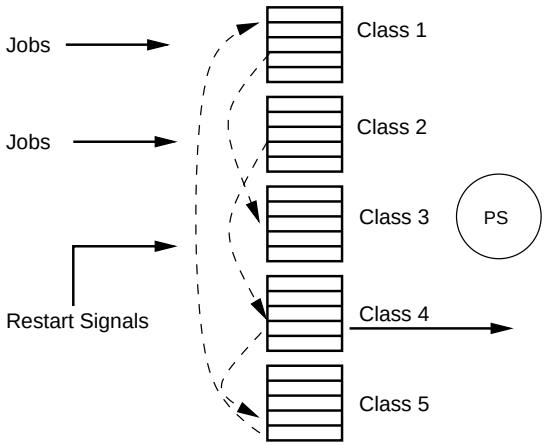
Third example: more complex service times

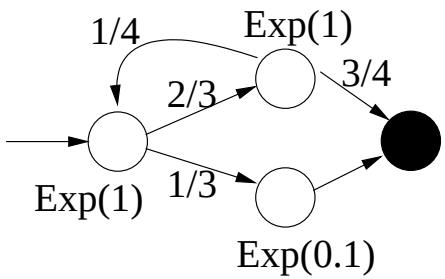




- Job arrivals to class 1 ($\lambda_1 = 0.004$) and class 2 ($\lambda_2 = 0.01$)
- Restart succeeds with prob 0.5 in phase 1 and 2, with prob 1 in phase 3

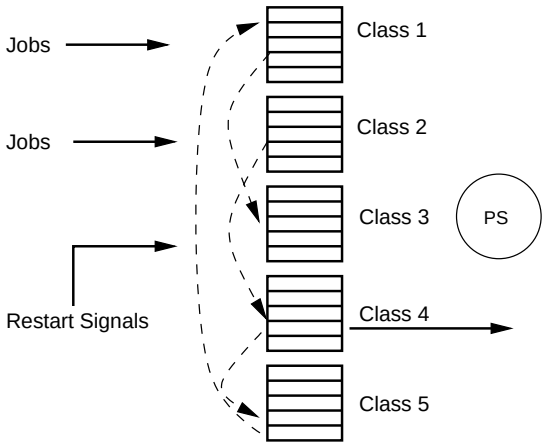
Third example: more complex service times

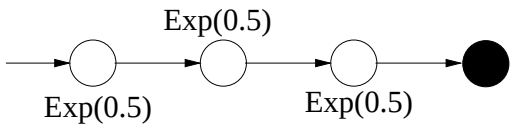




- Job arrivals to class 1 ($\lambda_1 = 0.004$) and class 2 ($\lambda_2 = 0.01$)
- Restart succeeds with prob 0.5 in phase 1 and 2, with prob 1 in phase 3
- In class 2 service time distribution is no APH, restart succeeds only in phase 3

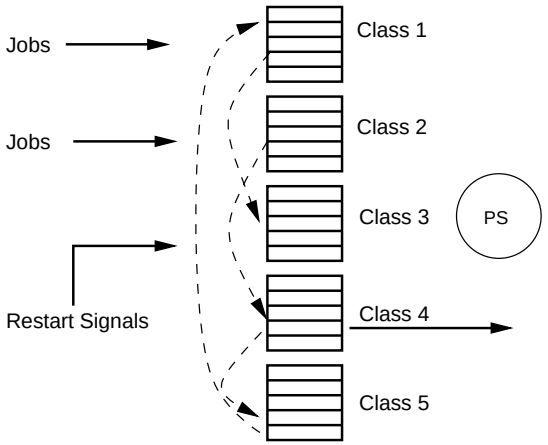
Third example: more complex service times

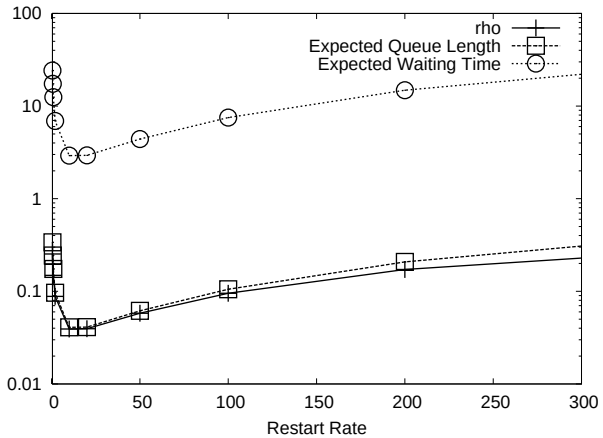




- Job arrivals to class 1 ($\lambda_1 = 0.004$) and class 2 ($\lambda_2 = 0.01$)
- Restart succeeds with prob 0.5 in phase 1 and 2, with prob 1 in phase 3
- In class 2 service time distribution is no APH, restart succeeds only in phase 3
- Class $3, 4, 5$ have Erlang(3) service time with $\lambda = 0.5$ and different restart success probs

Third example: more complex service times





- Job arrivals to class 1 ($\lambda_1 = 0.004$) and class 2 ($\lambda_2 = 0.01$)
- Restart succeeds with prob 0.5 in phase 1 and 2, with prob 1 in phase 3
- In class 2 service time distribution is no APH, restart succeeds only in phase 3
- Class $3, 4, 5$ have Erlang(3) service time with $\lambda = 0.5$ and different restart success probs
- Restart has strong positive effect

Conclusions

- G-networks can have several classes in several queues

Conclusions

- G-networks can have several classes in several queues
- Different type of job arrivals are possible - different clients

Conclusions

- G-networks can have several classes in several queues
- Different type of job arrivals are possible - different clients
- Signals can model restarts

Conclusions

- G-networks can have several classes in several queues
- Different type of job arrivals are possible - different clients
- Signals can model restarts
- Product-form solution, standard queueing measures

Conclusions

- G-networks can have several classes in several queues
- Different type of job arrivals are possible - different clients
- Signals can model restarts
- Product-form solution, standard queueing measures
- Great flexibility in service time distributions, Phase-type

Conclusions

- G-networks can have several classes in several queues
- Different type of job arrivals are possible - different clients
- Signals can model restarts
- Product-form solution, standard queueing measures
- Great flexibility in service time distributions, Phase-type
- Find optimal restart interval in most situations, i.e. restart reduces response time

Conclusions

- G-networks can have several classes in several queues
- Different type of job arrivals are possible - different clients
- Signals can model restarts
- Product-form solution, standard queueing measures
- Great flexibility in service time distributions, Phase-type
- Find optimal restart interval in most situations, i.e. restart reduces response time
- Modelling power for arrivals not fully exploited

Conclusions

- G-networks can have several classes in several queues
- Different type of job arrivals are possible - different clients
- Signals can model restarts
- Product-form solution, standard queueing measures
- Great flexibility in service time distributions, Phase-type
- Find optimal restart interval in most situations, i.e. restart reduces response time
- Modelling power for arrivals not fully exploited
- Formalism to study reliable protocols, rejuvenation, checkpointing, ...

Conclusions

- G-networks can have several classes in several queues
- Different type of job arrivals are possible - different clients
- Signals can model restarts
- Product-form solution, standard queueing measures
- Great flexibility in service time distributions, Phase-type
- Find optimal restart interval in most situations, i.e. restart reduces response time
- Modelling power for arrivals not fully exploited
- Formalism to study reliable protocols, rejuvenation, checkpointing, ...
- Need better tool support.

Thank you.