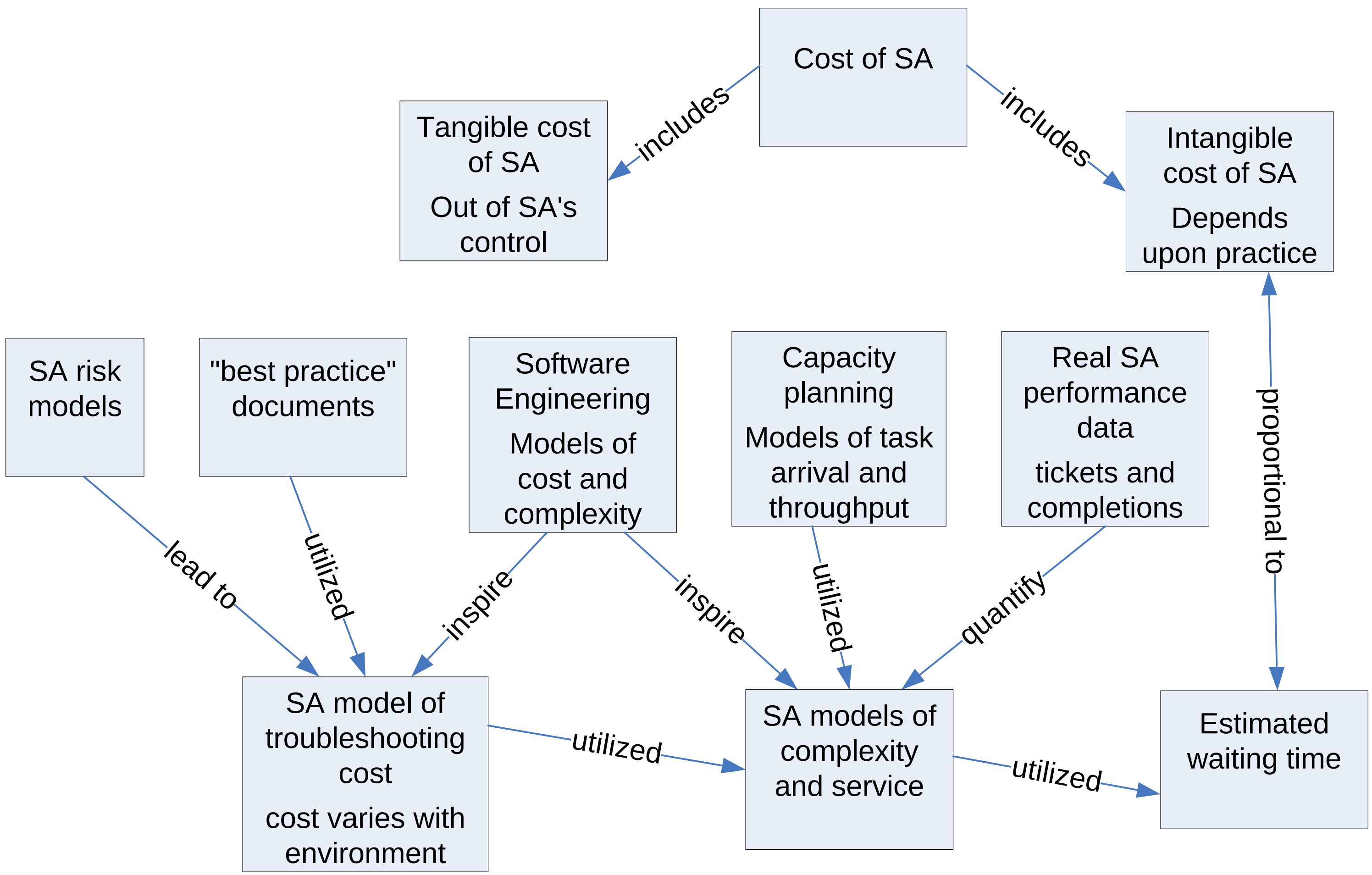
Toward a cost model for system administration

Alva Couch
Ning Wu
Hengky Susanto
Tufts University

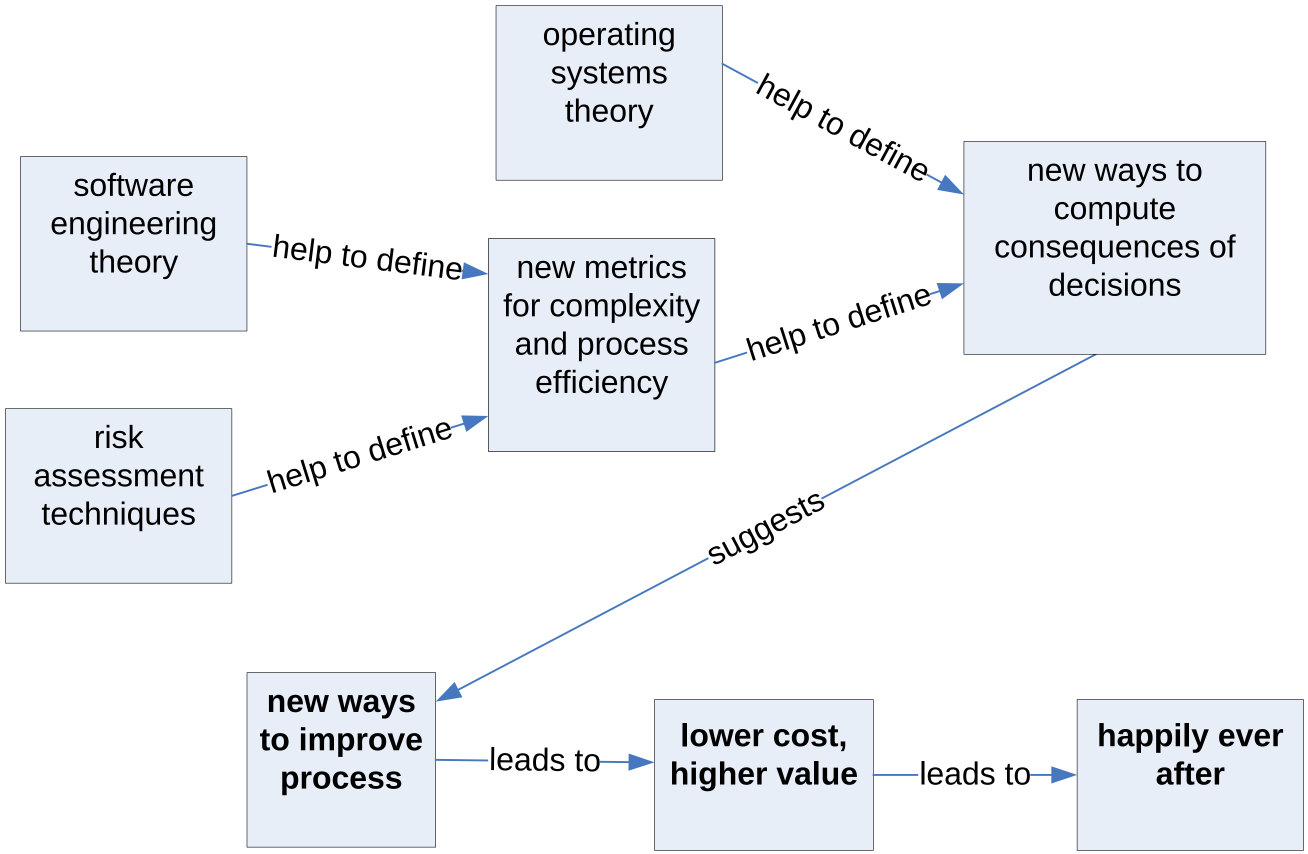
Executive Summary



LISA-2005
couch@cs.tufts.edu

Tufts University
Computer Science

System Administrator's Summary



"Best Practices"

- Cost the least
- Provide the most value
- via several intangibles
  - homogeneity
  - consistency
  - repeatability
  - documentation
  - etc.

Patterson's cost model

- Cost of downtime ≈ cost of revenue lost + cost of work lost.
- Patterson, "A simple model of the cost of downtime", Proc. LISA 2002
- Controversial: downtime cost is "intangible".
- Or is it?

“Best” is relative!

- Patching systems immediately causes more downtime than waiting for patches to stabilize.
- Cowan et al, “Scheduling the application of security patches for optimal uptime”, Proc. LISA 2002.

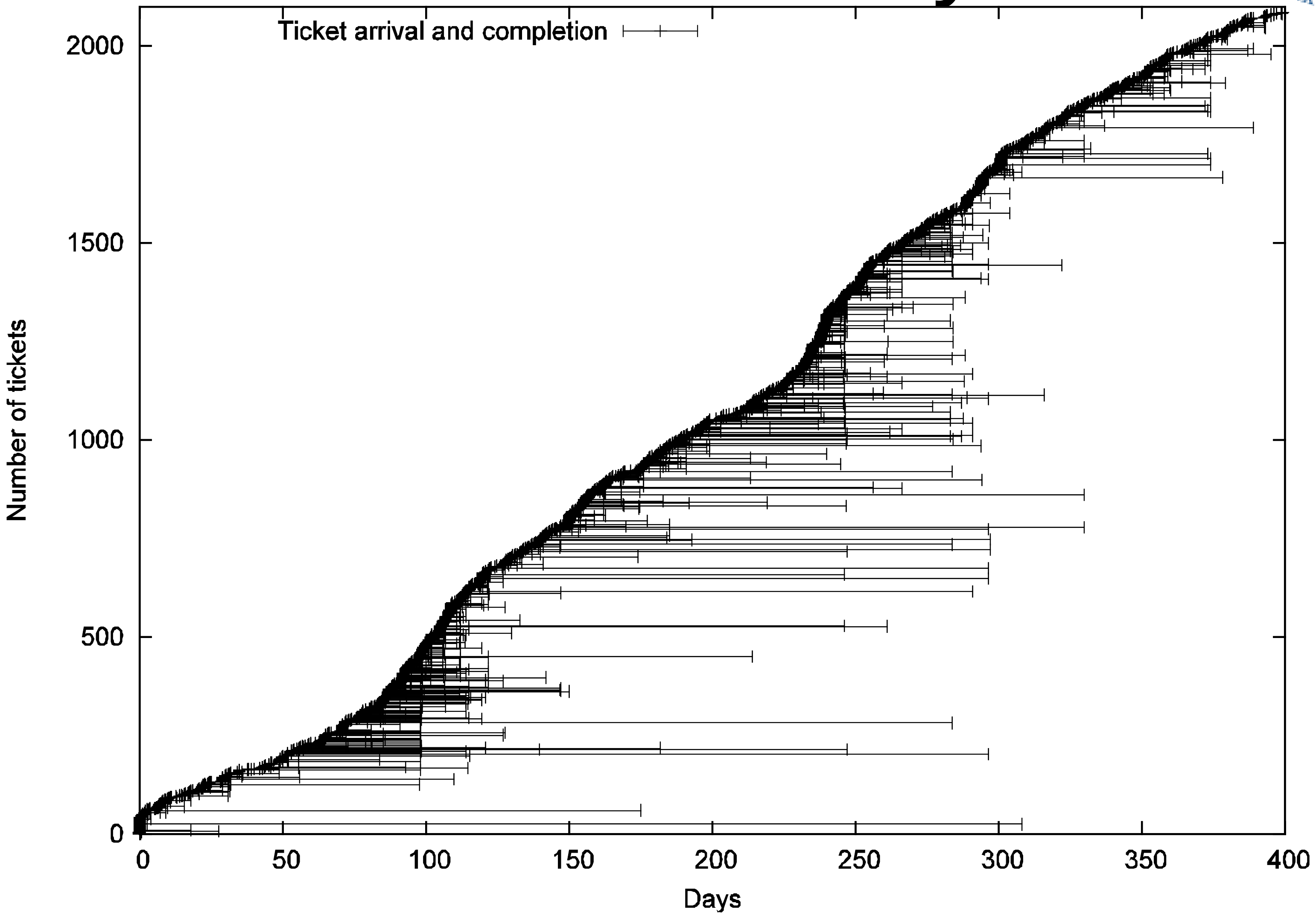
Time spent waiting

- Cost of system administration = cost of tangible assets + cost of intangibles
- For most SA's, cost of tangible assets is out of our control.
- Claim 1: **The intangible cost of system administration is approximately proportional to (cumulative) time spent waiting for responses to requests**

Learning from real data

- Data source: RT queue, Tufts ECE/CS.
- Data duration ≈ 400 days.
- What is the structure of real data?
- Is there any easy way to describe the schedule of ticket arrivals and service?

Ticket history



Measuring time spent waiting

- Time spent waiting is a function of
  - **arrival rate**: number of requests coming in
  - **service rate**: how fast requests can be processed
  - **number of “workers”** available
  - **number of “clients”** affected.
- Where
  - arrivals include reconfigurations and refits
  - rate is reciprocal of expected service time

Memory

- A process is **memoryless** if the next event does not depend upon the history of prior events.
  - memoryless arrivals: “Poisson process”

    λ = **arrival rate,** mean inter-arrival time = $1/\lambda$, standard deviation of inter-arrival times = $1/\lambda$.
  - memoryless service: “exponential service time”.

    μ = **service rate,** mean service time = $1/\mu$, standard deviation of service time = $1/\mu$.

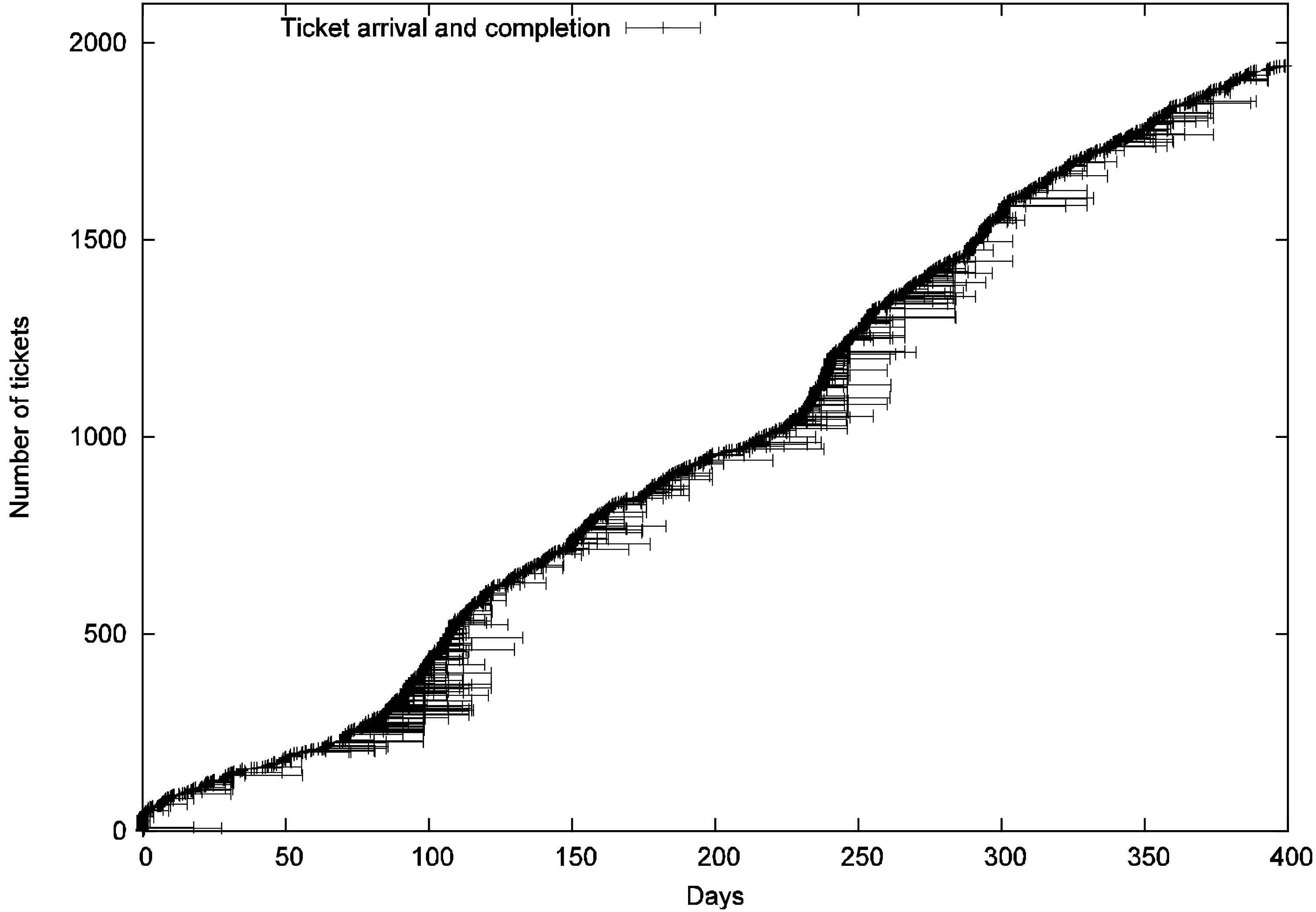
Memoryless is nice (but perhaps impractical)

- Memoryless arrivals: lots of identical customers behaving independently.
- Arrival processes with memory: bursty behavior, such as a virus infection, spam, or DDoS attack.
- Advantage of memoryless models: closed-form solutions to system performance (from capacity planning)

Multiclass systems

- Typical site has **multiple classes** of requests; some are more complex or take longer than others.
- At first glance, no exponential service times.
- Throw away long times (outliers); exponential service times emerge!
- **Claim 2: Documentation keeps requests from waiting indefinitely.**

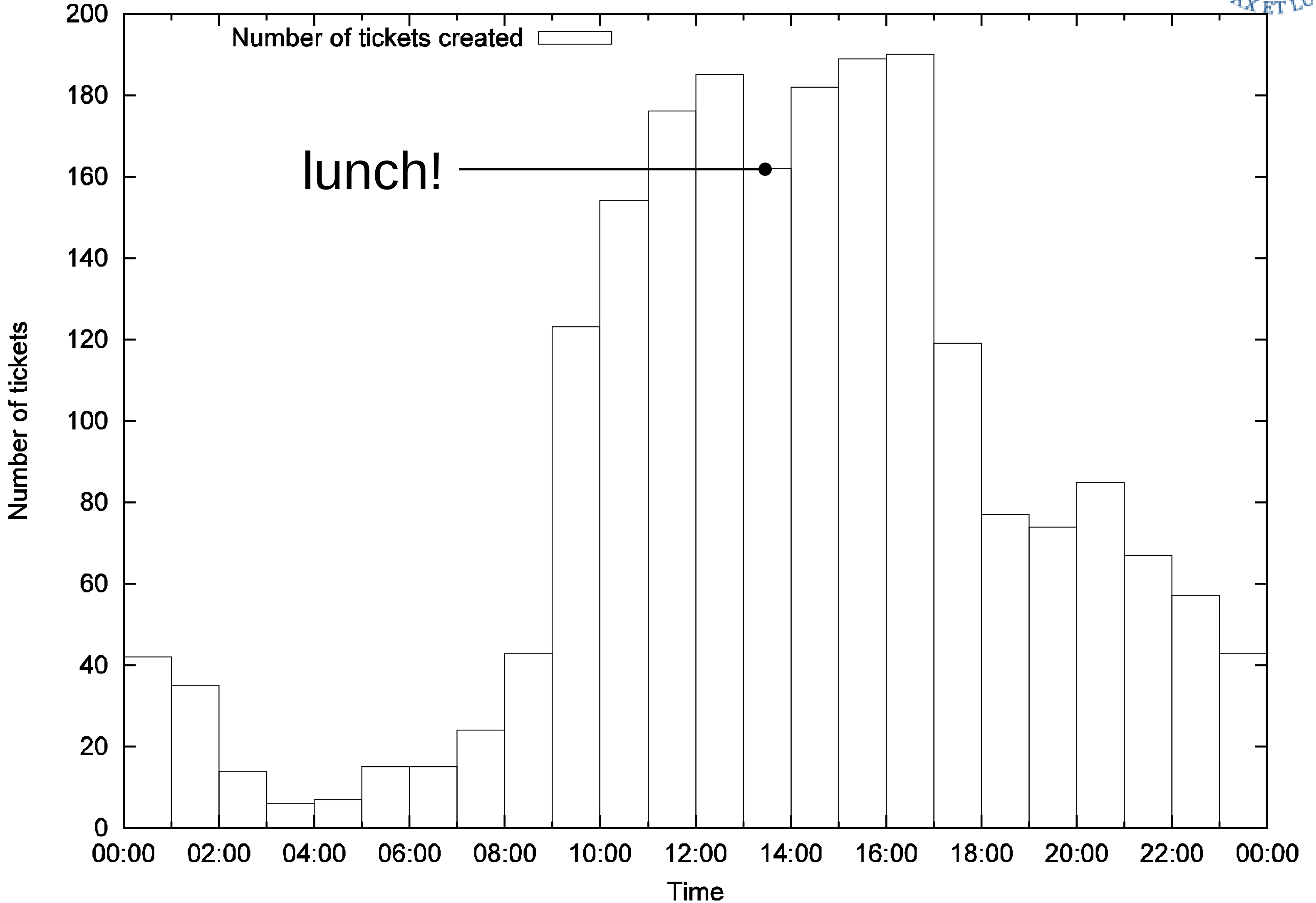
Tickets filtered



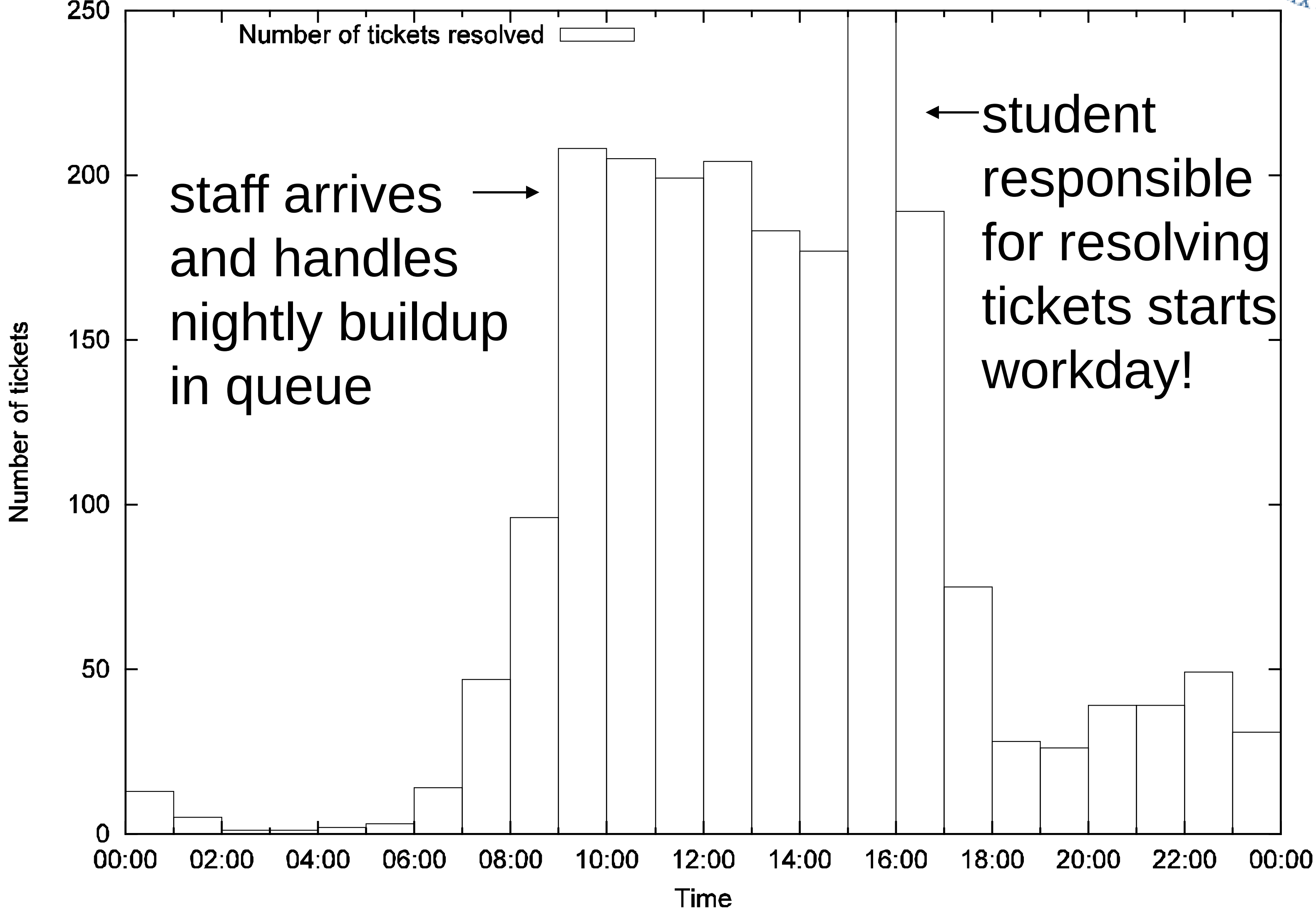
Quandary of arrivals

- At first glance arrivals aren't Poisson
- But (a month of struggling later!)
  - correct for DST
  - sample over one-hour intervals
  - correct sampling for sparse event frequency
  - skip holidays
- And each **hour** exhibits a roughly Poisson arrival rate!

Ticket creation



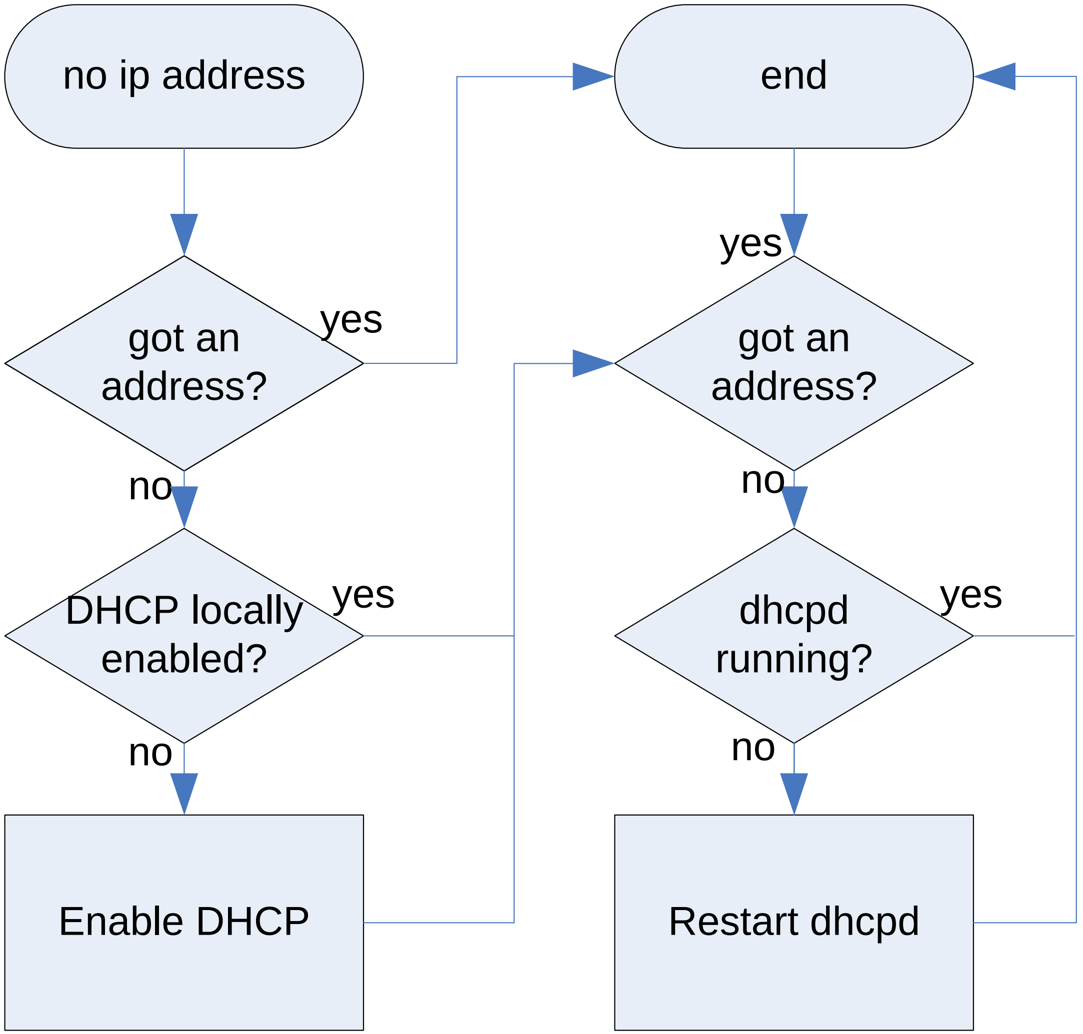
Ticket resolution



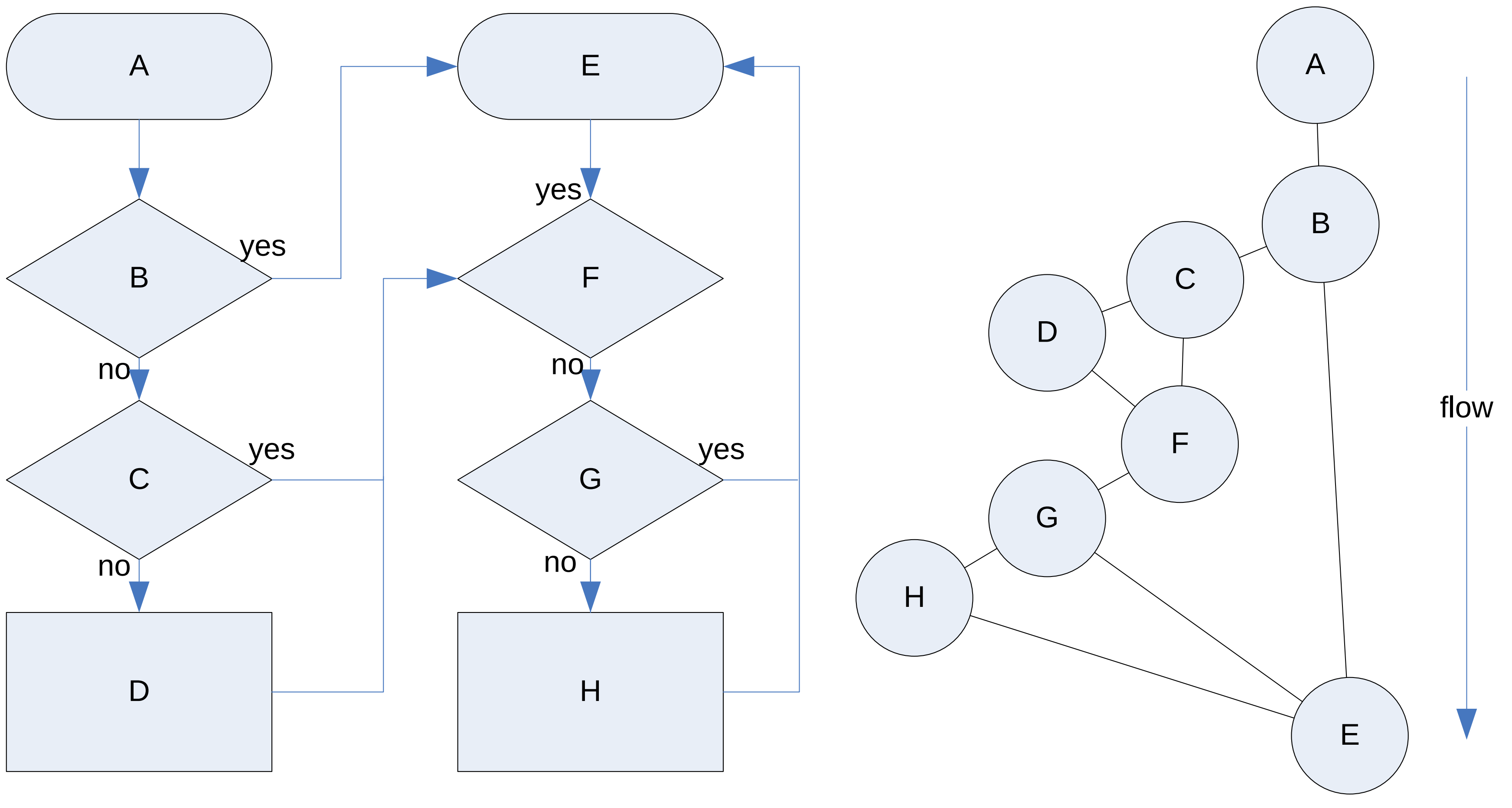
Quantifying time spent waiting

- Our data shows that most requests are actually accomplished at our site in (statistically) comparable times.
- How does one estimate the time needed for a particular request?
- One example: troubleshooting chart.

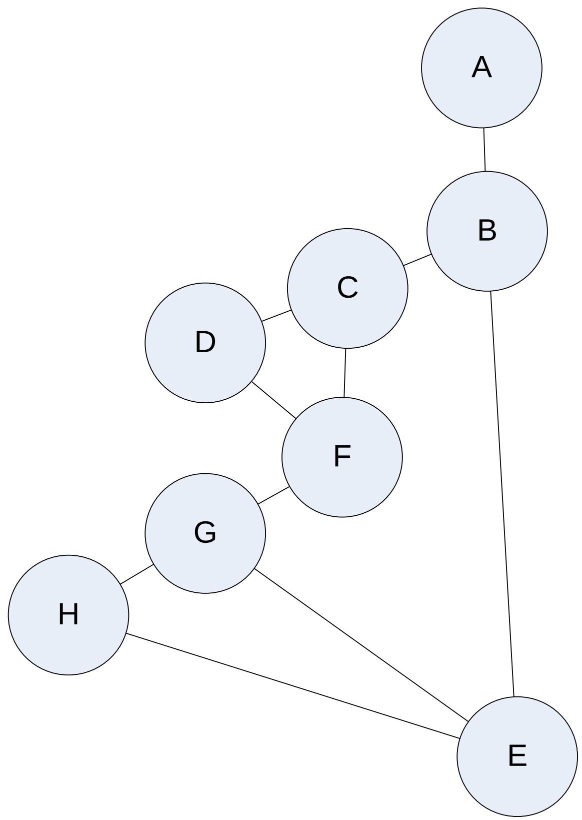
Simple troubleshooting chart

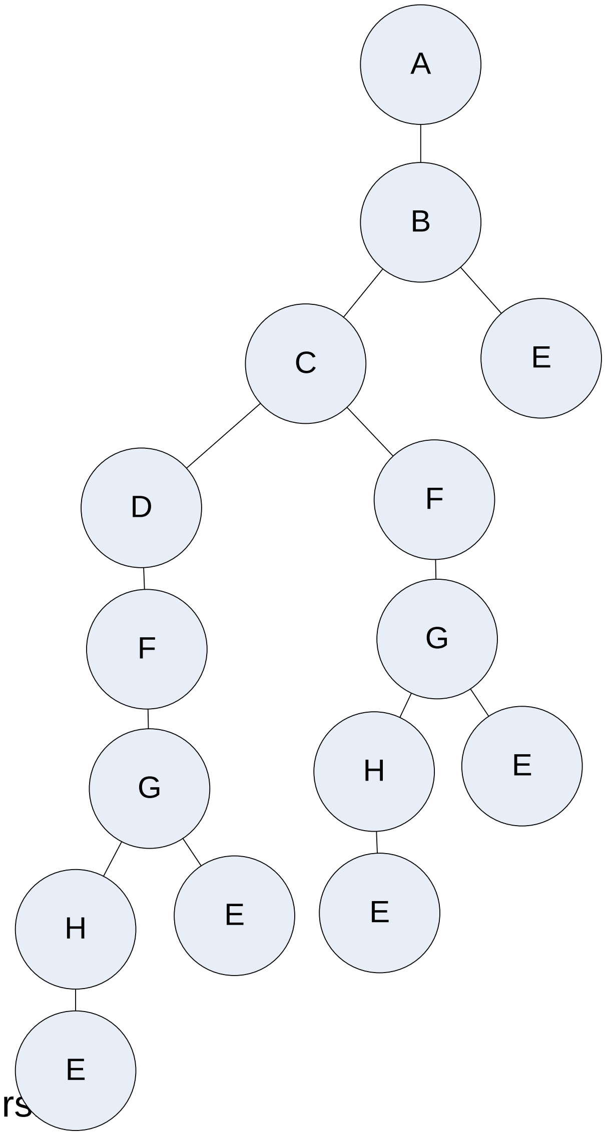


Convert to program graph

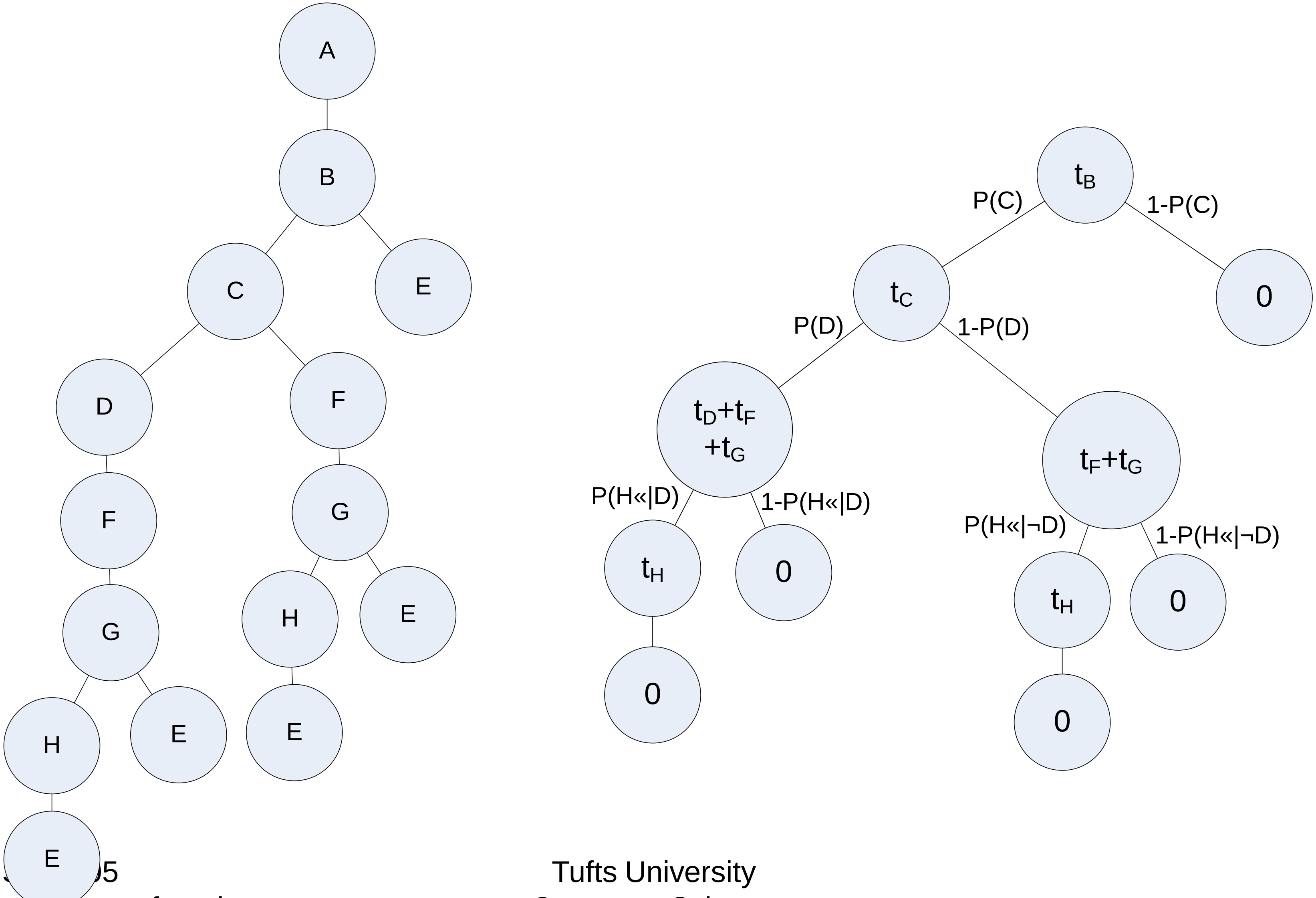


Convert from graph to tree



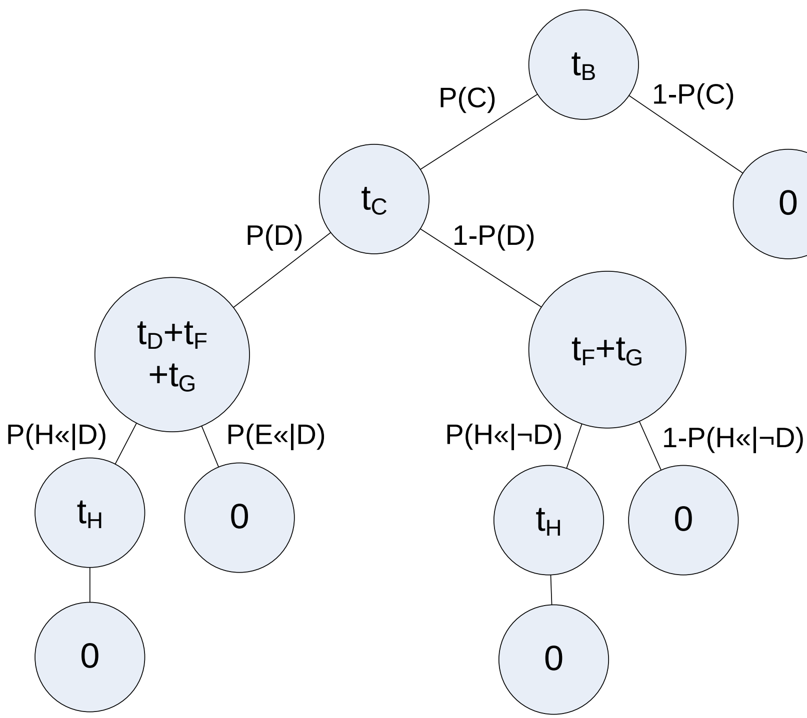


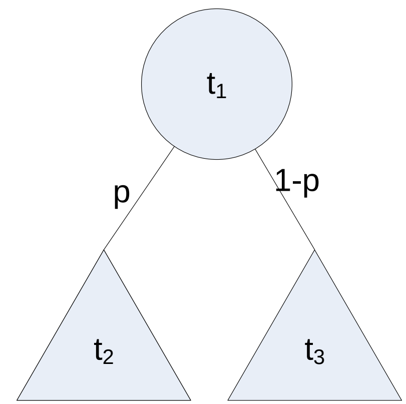
Collapse to decision tree



Compute expected value

expected wait = t_B+P(C) [
t_C+P(D)[t_D+t_F+t_G+P(H«|D)t_H)+(1-P(D))(t_F+t_G+P(H«|¬D)t_H]
]





expected wait = t1 + pt_2+(1-p)t_3

Notes on the decision tree

- Times t_X describe the *capabilities of administrative staff.*
- Probabilities P(Y) describe the *site's characteristics and the likelihood of failures*.
- P(H«|D): probability of H happening given that D happened *in the past*
- [temporal conditional probability; not Bayesian; Bayesian identities don't hold! Another month of suffering to figure this out!]

Application: should I check the DHCP server or client first?

- Answer: depends upon site characteristics.
- If the likelihood is that there is a problem with X, should check X first.
- Consequences of incorrect choice: *increased cost.*
- Humans *automatically compensate* for poor troubleshooting order.
- Claim 3: **Best practices are relative to site and staff capabilities.**

Bang!

- The preceding method is “white box”; it measures the practice directly.
- Applying the preceding argument for a non-trivial troubleshooting chart results in an **exponential explosion** in chart complexity.
- How do we deal with huge charts or complex processes?
- Answer: “black box” **estimation**.

Estimators from Software Engineering

- Time for service is approximately a function of the number of branches in a troubleshooting chart.
- Number of branches is approximately a function of heterogeneity/diversity of site and services provided.
- So if we quantify diversity/complexity of service environment, we can estimate service time.
- “Function points”: a way of quantifying complexity of service.

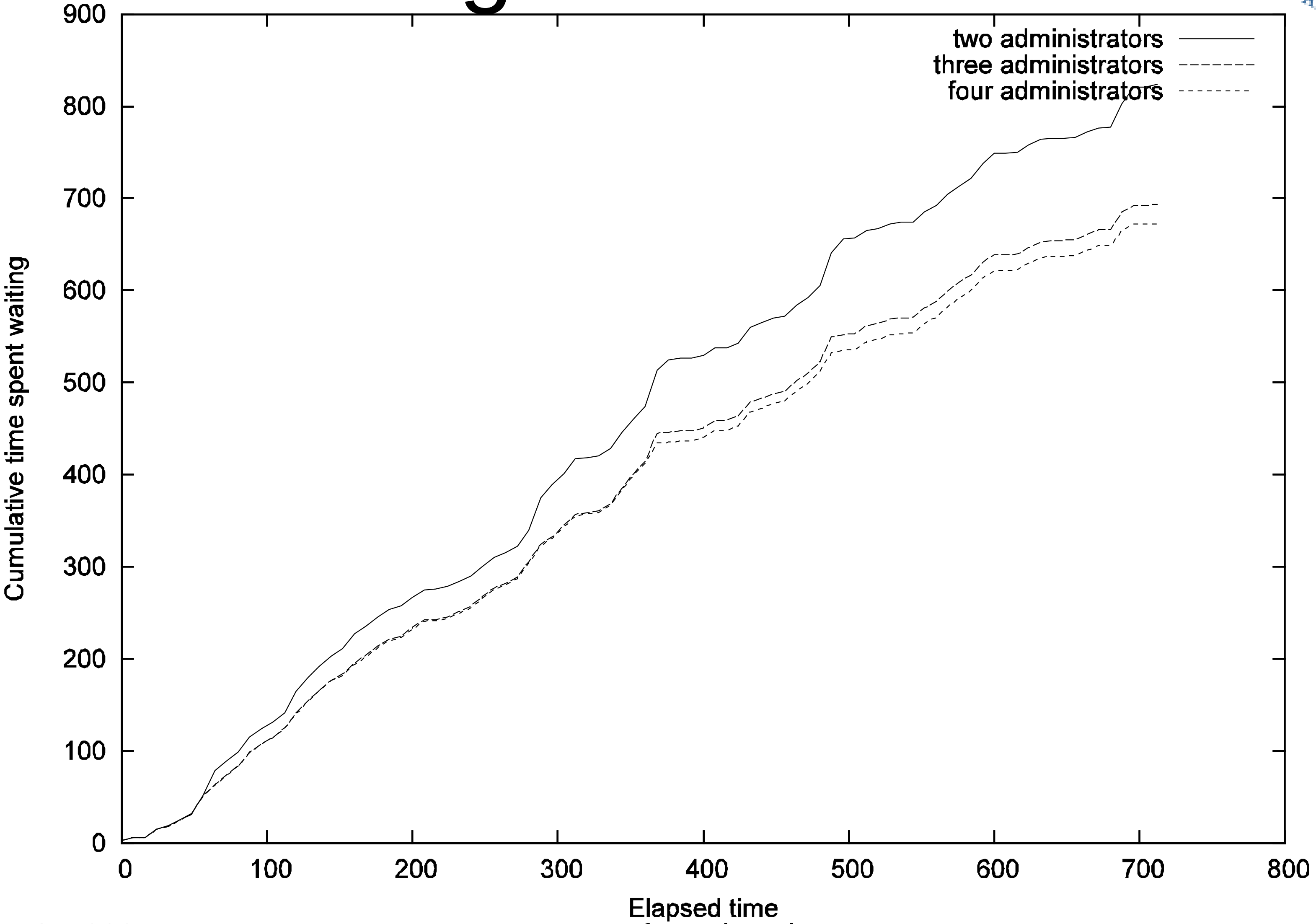
Non-product systems

- We understand a great deal about "product systems" in which components act independently.
- System administrators are a non-product system; they communicate and interact with **each other.**
- Best way to estimate behavior of non-product systems: **discrete event simulation.**

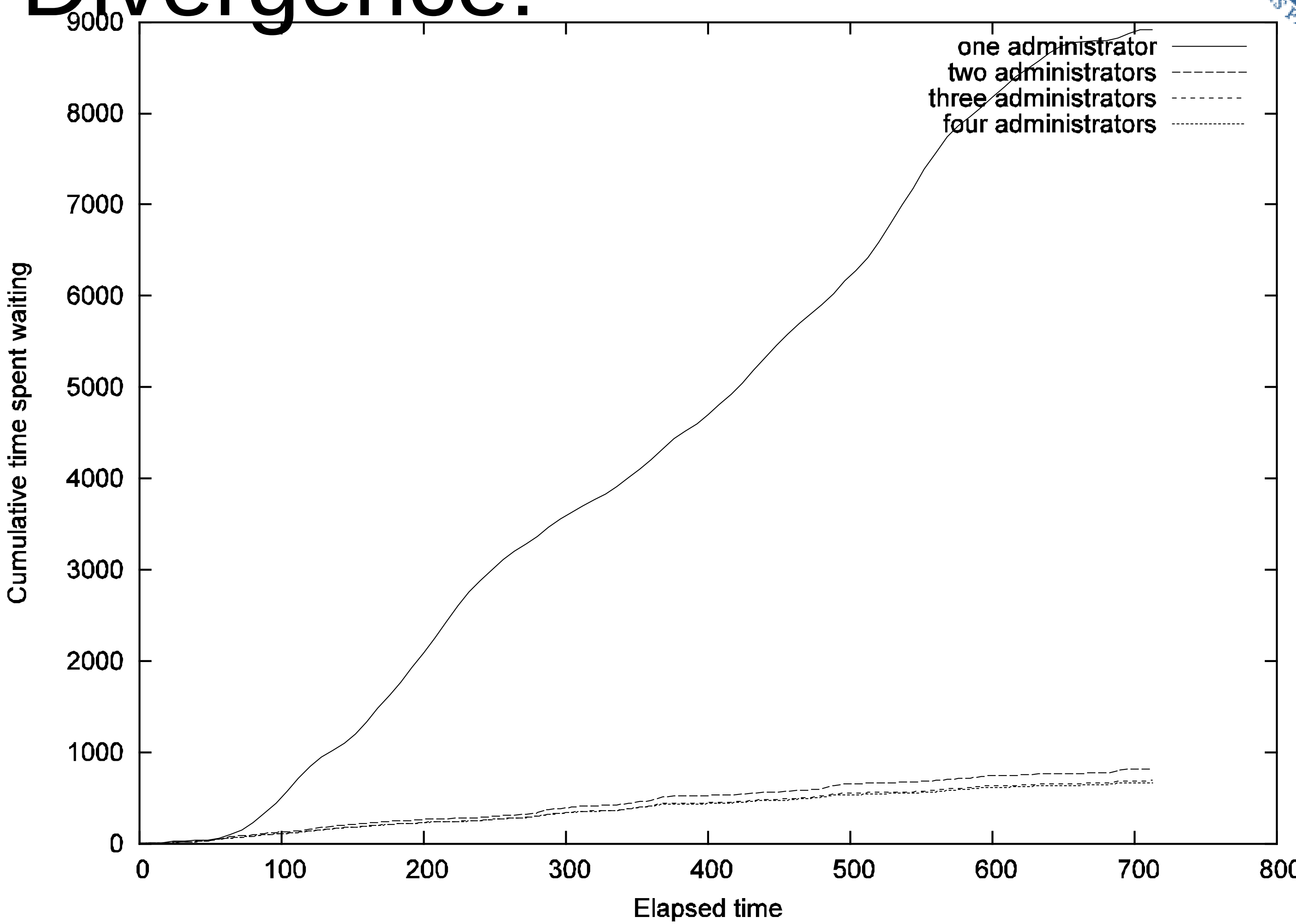
A simple simulation experiment

- Assume c administrators, four classes of service (from extremely short to extremely long service times), independent arrival rates for classes.
- Theory: a single class system is stable if $\boldsymbol{\lambda/c\mu<1}$ and diverges to infinite wait time otherwise.
- What happens when a multi-class system approaches the saturation point?

Diminishing returns

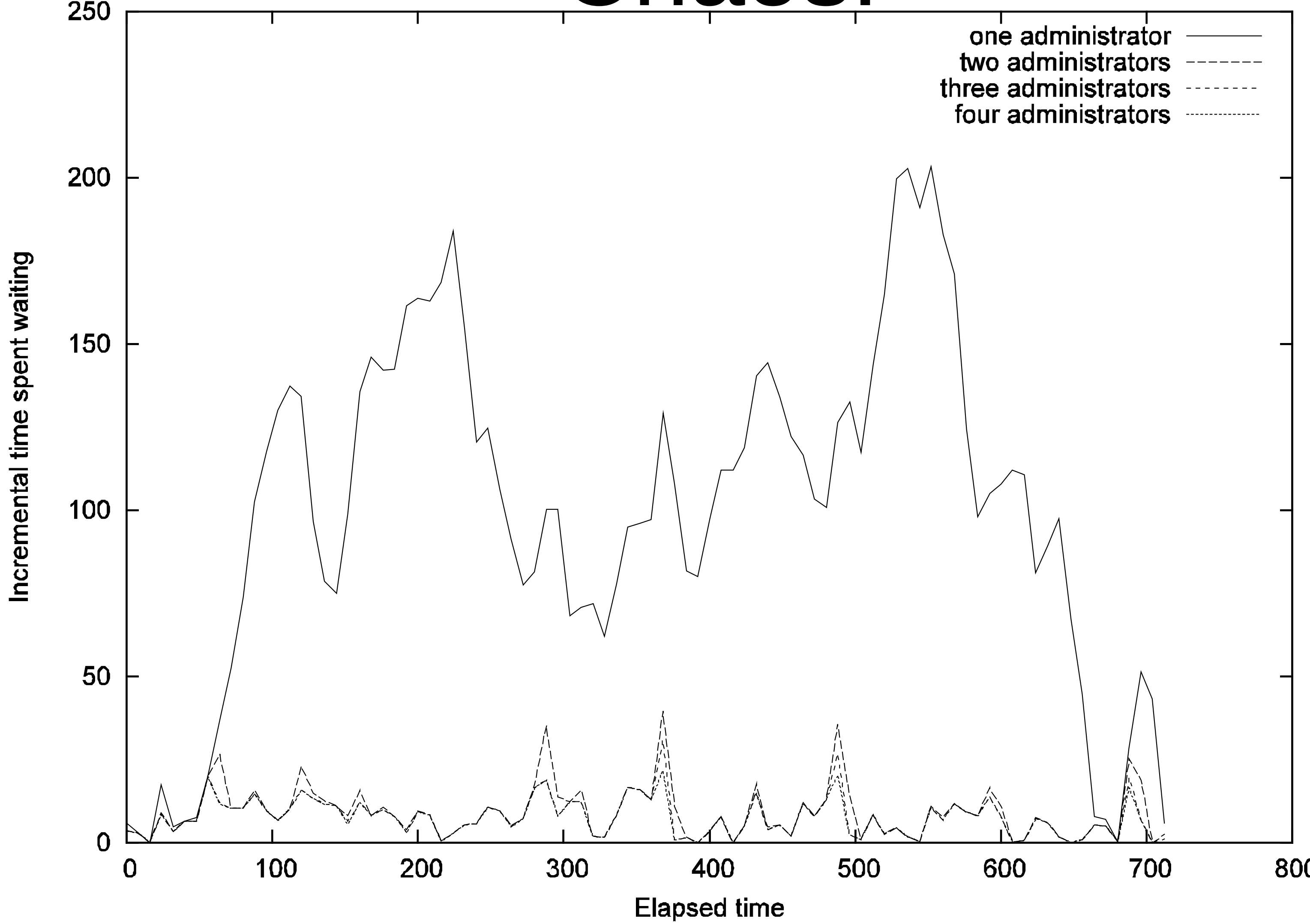


Divergence!



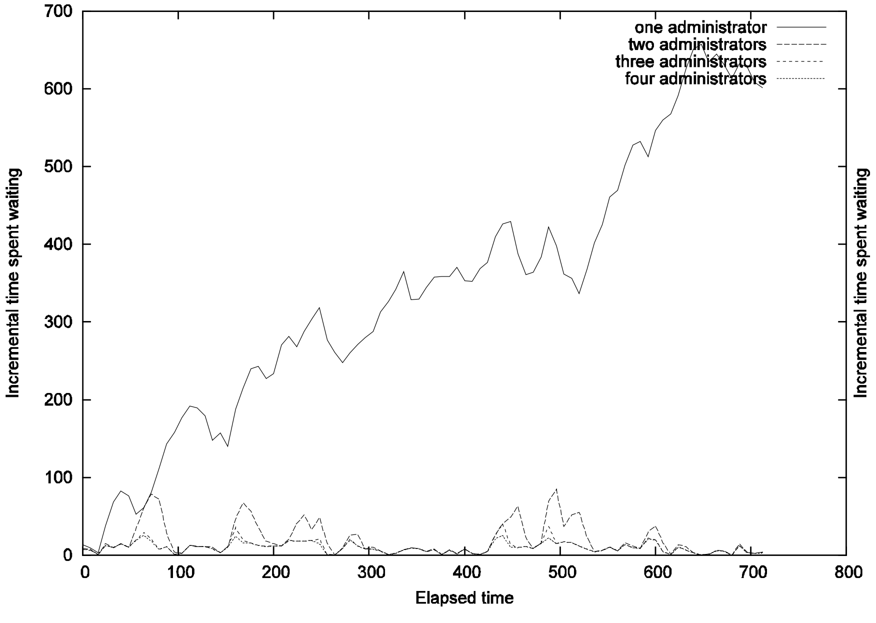
one administrator
two administrators
three administrators
four administrators
Cumulative time spent waiting
Elapsed time
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
0
100
200
300
400
500
600
700
800

Chaos!

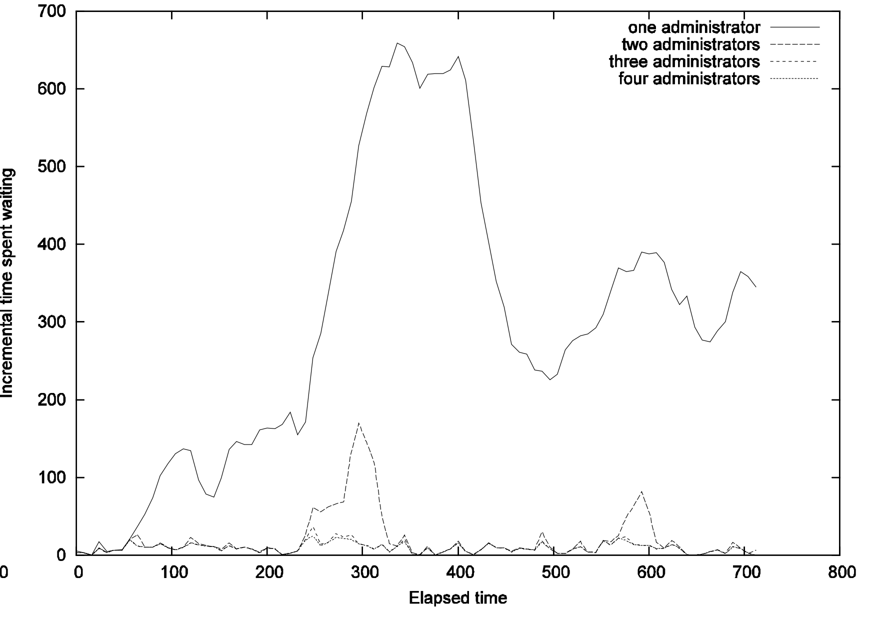


one administrator
two administrators
three administrators
four administrators
Incremental time spent waiting
Elapsed time
0
50
100
150
200
250
0
100
200
300
400
500
600
700
800

Running near the edge



arrivals spread out



bursty arrivals

events in a burst, versus events spread out!

Summary

- cumulative service time ≈ intangible cost of operations
- computable from practice graph: function of staff expertise and site composition.
- estimable from guesses for branch depth and task length for each task.
- total effect estimable via discrete event simulation.

Conclusions

- We can estimate the cost of practice by indirect methods.
- Best practices are *always* site relative!
- Running near absolute capacity causes chaotic increases in wait time.

What's next?

- Simulation studies of particular aspects of the practice:
  - communication vs. documentation,
  - scripting vs. cfengine
- Quantification of function point models
  - various sizes and kinds of sites.
  - complexities of kinds of service.
- Effects of human learning
  - Insignificant for repetitive tasks.
  - Significant for one-time tasks.

Epilogue

- More questions than answers:
  – How can we best use this as a planning tool?
  – How much can we trust it?
  – How to fill in gaping holes in knowledge?
- The potential:
  – better/cheaper/more valuable administrative practices.
  – Ability to ask cheap "what if" questions with reasonable estimates of task complexity.
  – better understanding of critical capacity.
  – happily ever after.

Questions?

Alva Couch (couch@cs.tufts.edu)

Ning Wu (ningwu@cs.tufts.edu)

Hengky Susanto (hsusan0a@cs.tufts.edu)

Tufts University Computer Science

Medford, MA 02155

Note: we plan to make the discrete event simulator **open source** at some future time after we clean up the user interface.