The Joy of Queueing

Computer Science In Practice ~

Prof. Liam Murphy & Prof. John Murphy

Outline

- The Take-Home Message
- Random Flows and Stability in Queues
- Little’s Law: the most basic queue equation
- Examples of Queues in Computer Science
- Complications
- Conclusions and questions

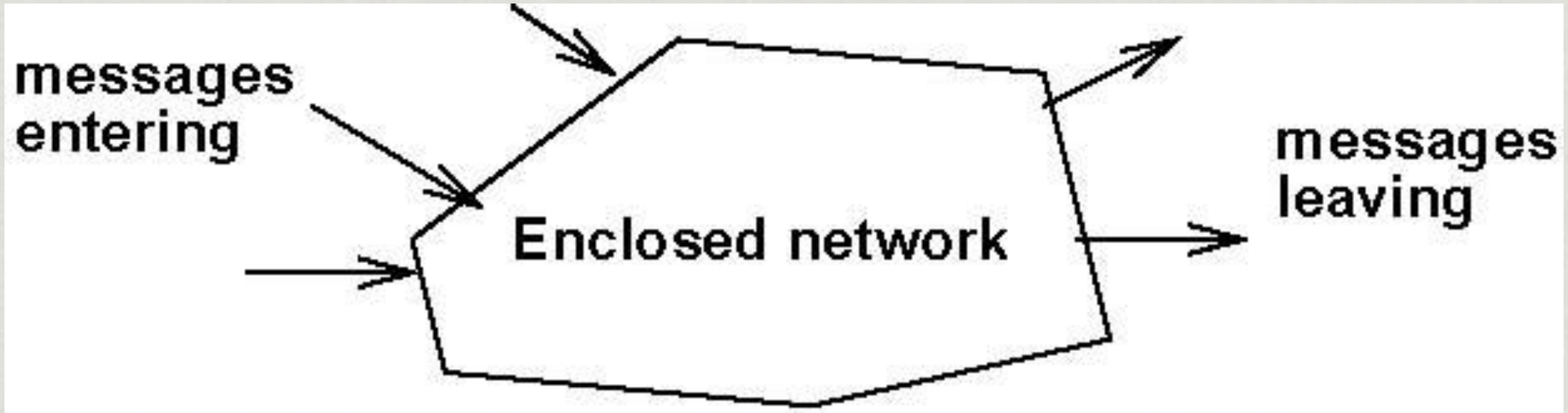
Take Home Message

- QUEUEING IS EVERYWHERE
- The **same approach(es)** can be applied to solve queueing problems in **very different areas**
  - Most real queueing systems cannot be solved exactly
    - Usually only approximate solutions can be found…
  - …but even these can give vital – and surprising – insights

Introduction

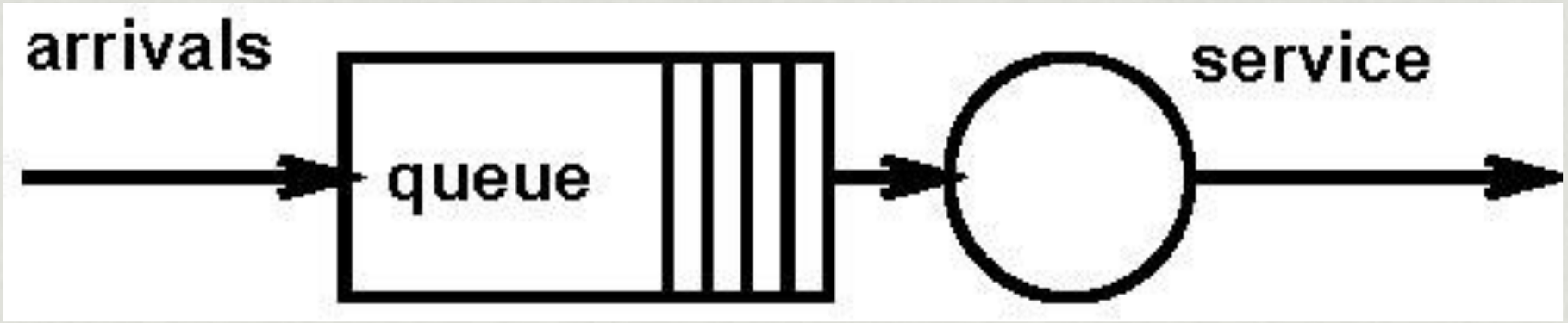
- When you have to wait in a bank, or a supermarket check-out line, or at the doctor's office, or to get into a nightclub…
- When you drive/cycle in the city, fly on a plane, take the bus…
- When you browse a website, send an email, make a phone call, …
- …then you're a **customer** in a **queueing system**

Queueing System 1



- A customer presents a **job** to be carried out by the system
- No jobs are created or destroyed in the system
  - Jobs can flow in or out of the system across the boundary, or can be stored in the system

Queueing System 2



- Arrival process: **average rate λ**
- Queue process: method, buffer size
- Service process: **average rate μ**

Random flows 1

- If the arrival pattern of customers to the queue was **predictable**, and the amount of time to serve them was also **predictable**, then the queue could be sized to serve all customers with no waiting
  - However, life is rarely this simple :^)
- In reality the customers usually arrive in some ***random*** manner, and the amount of service they require is also ***random***
  - Good news: “random” doesn’t mean “cannot be analysed”

Random flows 2

- A queueing system can be in a **transient** or a **stable** state –
  - Transient: starting up, change in properties, …
  - Stable: when the properties have settled down

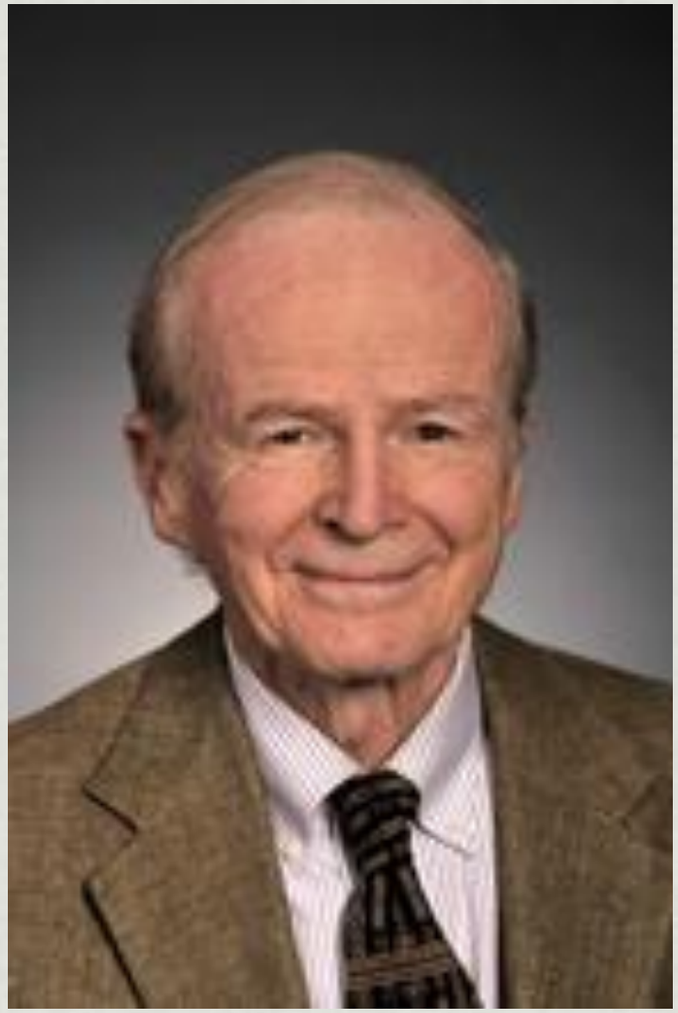
    …but, this does **not** mean things are predictable!

- Stability:

  Average input rate = Average output rate

Little's Law

- Simple, yet applies to nearly all practical queueing systems
- "The average number of jobs stored in a system is equal to the product of the average arrival rate of jobs to the system and the average time these jobs spend in the system"
- Mathematically speaking, **N = λ T**
  - N = average number of jobs in the system
  - T = average time a job spends in the system



John Little (1961) ***then at Case Western Reserve University***

Little's Law: example

- Suppose you stand at the only door in or out of a nightclub, and you measure the rate at which people are allowed in (which is also the rate at which other people leave) to be 2 people per minute, on average over the period when you're observing them.
- You also ask each person leaving how long they spent in the nightclub – let's say the average of these times is 30 minutes (and that they're telling the truth!)
- How many people are inside, on average? Call it N.
- Answer: $N = 2 \times 30 = 60$ people.

Queueing results 1

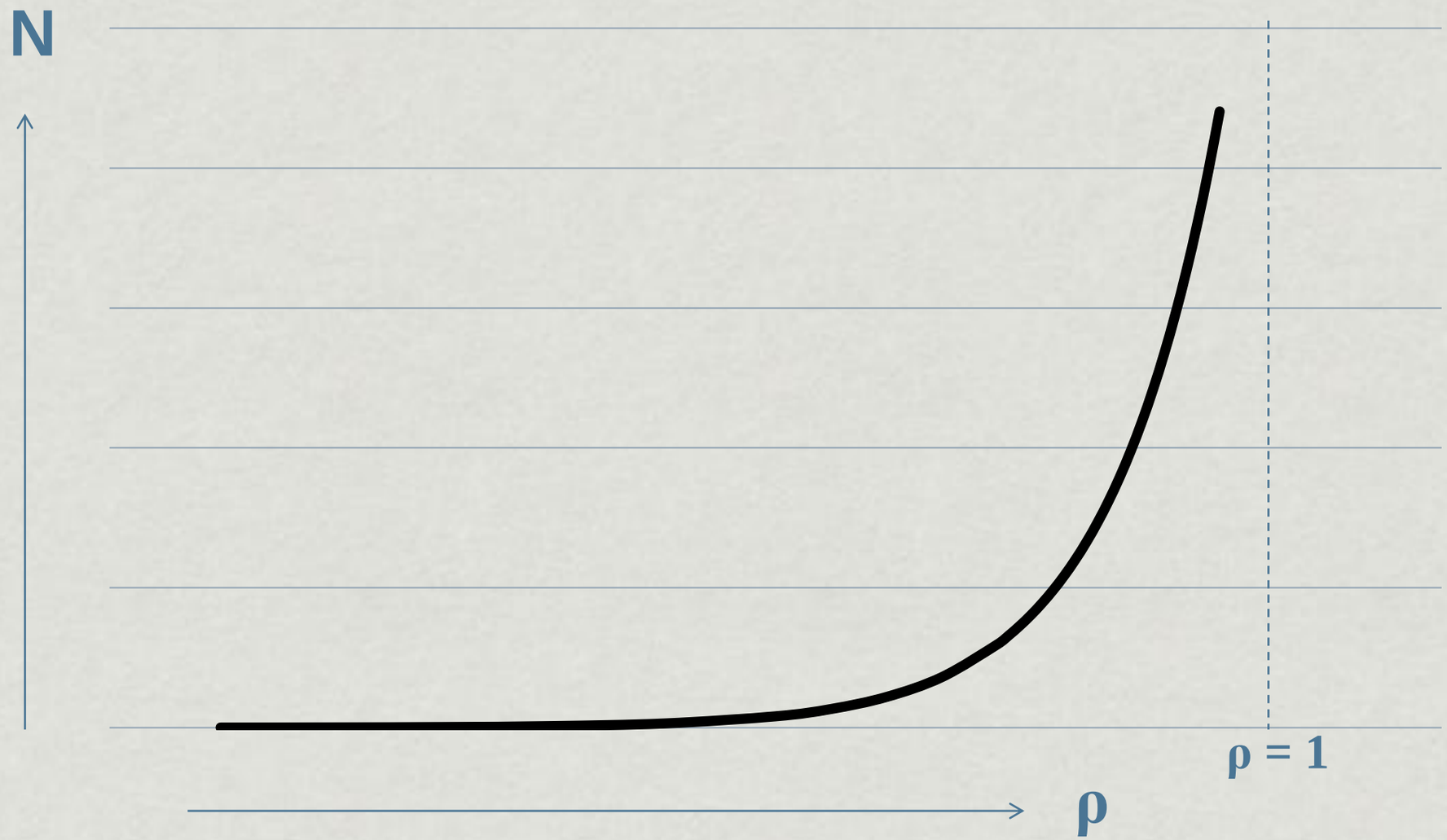
- For a common type of queue, the average number of jobs in the system is $N = \rho / (1- \rho)$

- where $\rho = \lambda/\mu$ **must** be < 1

- i.e. under assumptions which are often true in practice, you get a non-linear curve…

Queueing results 2

- a non-linear curve, not a linear relationship like you might (naively) expect…

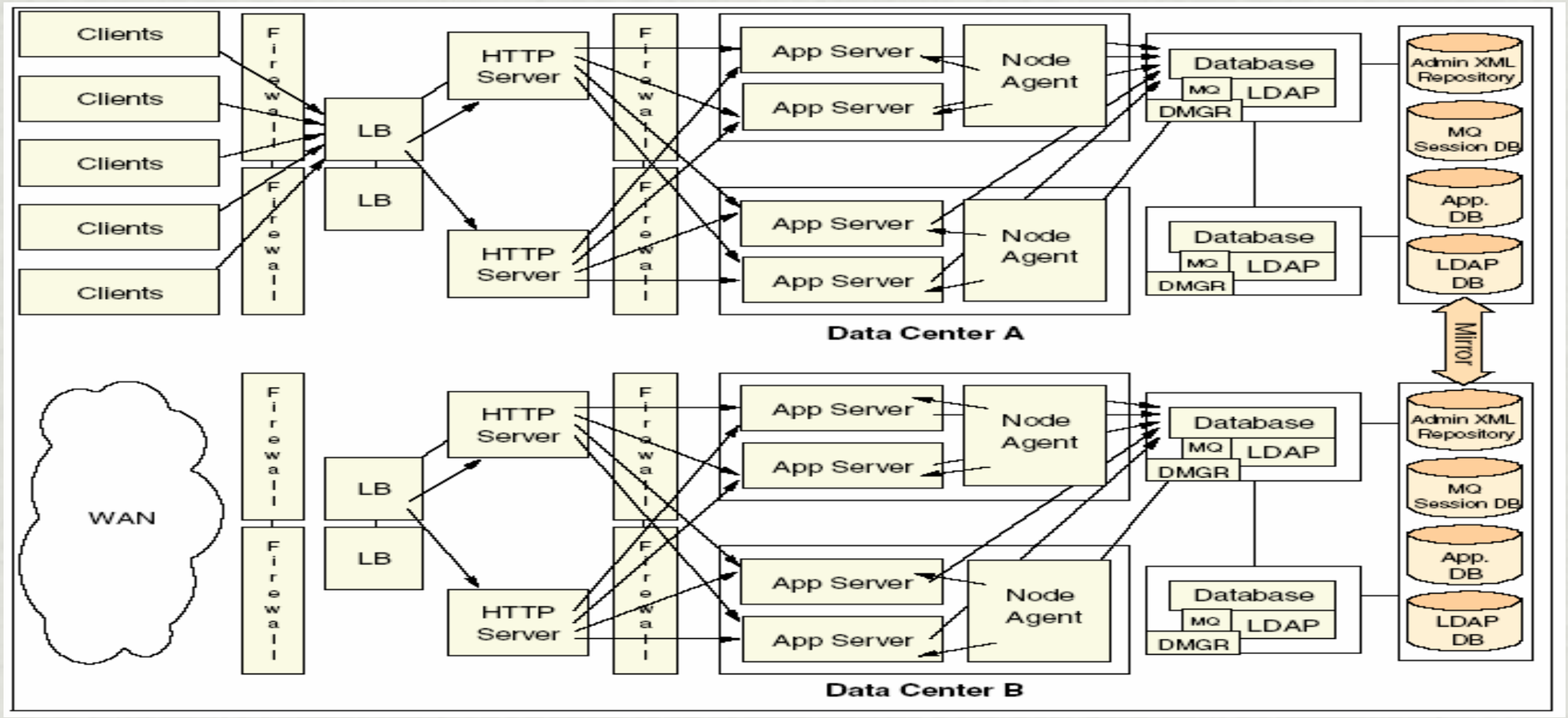


Queueing results 3

- More on the M/M/1 queue: the assumed arrival pattern and service rate are often true in reality
- Originally applied to telephone network with Erlang B and Erlang C tables

Agner Krarup Erlang (January 1, 1878 – February 3, 1929) was a Danish mathematician, statistician and engineer, who invented the fields of traffic engineering and queueing theory. Erlang created the field of telephone networks analysis. His early work in scrutinizing the use of local, exchange and trunk telephone line usage in a small community, to understand the theoretical requirements of an efficient network, led to the creation of the Erlang formula, which became a foundational element of present day telecommunication network studies *(from Wikipedia)*

Queues in Computer Science



- Application server or web server
- This is to show that queueing really can be applied to CS problems!

More about queueing

- ***Different pattern of arrivals and/or service.*** The pattern could be deterministic, or exponential, or something else.
- ***Finite buffer, have to turn customers away.*** Then we have arrivals that enter the system, and ones that are only offered but get rejected. Unstable?
- ***Faster server possible?*** Often not possible to speed up the server, so need multiple servers. Only useful with multiple customers.
- ***One overall buffer, or one buffer per server?*** With servers we could have a single queue for each, or an overall queue for the lot.

Complications in queueing

- **Priorities** (some customers are more important than others)
- **Pre-emption:** your service can be interrupted, then later it is resumed
- A **network** of queues: now the simplifying assumptions don’t hold

Take Home Message

- QUEUEING IS EVERYWHERE
- The **same approach(es)** can be applied to solve queueing problems in **very different areas**
  - Most real queueing systems cannot be solved exactly
    - Usually only approximate solutions can be found…
  - …but even these can give vital – and surprising – insights

Questions I

- List different types of queues that you come across in everyday life: some with a single server single queue, some with multiple servers (different queue types)
- Can you think of queues that have priorities? How does this affect the system?
- Can you think of any queues that might exist in a technical setting? (mobile phones, web systems)
- What sort of time scales are involved in the queues that you have outlined here?

Questions II

- What are the main features of a function that has a shape of $1/(1-\rho)$?
- What are the main differences between a system that allows people to queue (for ever) and one that limits the queue size to a maximum value?
- What happens if you have multiple servers working in a system and you allow people to queue (for ever) or limit the number?

Readings

- The definitive “bible” for queueing is by Leonard Kleinrock from 1975 (has over 10,000 citations)
- Queueing Systems. Vol 1: Theory
- Queueing Systems. Vol 2: Computer Applications

- **Professor Leonard Kleinrock is Distinguished Professor of Computer Science at UCLA. He developed the mathematical theory of packet networks, the technology underpinning the Internet, while a graduate student at MIT in the period from 1960-1962. *The birth of the Internet occurred in his UCLA laboratory* (3420 Boelter Hall) when his Host computer became the first node of the Internet in September 1969 and it was from there that he directed the transmission of the first message to pass over the Internet on October 29, 1969.**

More Readings…

- There are many online calculators for queueing problems, either in code (C, Java) or in spreadsheets
- If you search for “queueing” in Google Scholar, you will see a very large number of papers have been published on this topic!