On Some of My Simple Results

Leonard Kleinrock
Professor, UCLA Computer Science Dept.

Mobicom
September 8, 2014
1. Data Traffic is Bursty and Asynchronous

1960’s
The Problem with Bursty Asynchronous Demands

• You cannot predict exactly *when* they will demand access
• You cannot predict *how much* they will demand
• Most of the time they *do not need* access
• When they ask for it, they want *immediate* access!!
Conflict Resolution of Simultaneous Demands

• **Queueing:**
  - One gets served
  - All others wait

• **Splitting:**
  - Each gets a piece of the resource

• **Blocking:**
  - One gets served
  - All others are refused

• **Smashing:**
  - Nobody gets served!

A queueing system is a perfect resource sharing mechanism.

It serves whatever work has arrived.
How Fast Can You Serve?

- Most queueing systems consider that the “server” can only work at the rate of 1 sec/sec
How Fast Can You Serve?

- Most queueing systems consider that the “server” can only work at the rate of 1 sec/sec.

\[ T = \text{Response Time} \]

\[ \lambda = \text{Arrivals/sec} \]

\[ \bar{X} = \text{Avg Svc Time (sec)} \]
How Fast Can You Serve?

- Most queueing systems consider that the “server” can only work at the rate of $1 \text{ sec/sec}$
- Now replace humans with data technology

$T = \text{Response Time}$

$X = \frac{1}{\mu}$ (sec)

$\bar{X} = \text{Avg Svc Time (sec)}$

$\bar{X} = 1/ \mu C$ (sec)
The Basic M/M/1 Equation

\[
T = \frac{1}{1 - \rho} \quad \text{or} \quad T = \frac{\rho / \lambda}{1 - \rho}
\]

\[
\lambda = \text{Arrival rate (msg/sec)}
\]
\[
1 / \mu = \text{Avg No. of bits/msg}
\]
\[
C = \text{Capacity (bits/sec)}
\]
\[
\bar{x} = 1 / \mu \ C \quad \text{(sec)}
\]
\[
\rho = \lambda \bar{x} = \lambda / \mu \ C
\]

Now let's scale it up!
2. Economy of Scale

1960’s
Double the Throughput

$\lambda \\
\frac{C}{C} \\
T = \text{Response Time} \\
\frac{1}{\mu} = \text{bits/msg}$

Double the Capacity

$2\lambda \\
\frac{2C}{C} \\
\frac{1}{\mu} = \text{bits/msg}$
The Economy of Scale

• If you scale up throughput and capacity by some factor,
  
  \textit{then you reduce response time by that same factor.}

• If you scale capacity more slowly than throughput while holding response time constant,
  
  \textit{then efficiency will increase (and can approach 100%).}

• \textbf{If fact, you can improve all three!}
Key Tradeoff:
Response Time, Throughput, Efficiency

\[ \rho = \frac{\lambda T}{1 + \lambda T} \]

- Constant Response Time
- Throughput Increasing
- Efficiency Improving

\[ T = \frac{\rho / \lambda}{1 - \rho} \]

Response Time Improving
Throughput Increasing
Efficiency Improving

- Constant Efficiency
- Response Time Improving
- Throughput Increasing
- Constant Efficiency

Efficiency \%
0 20 40 60 80 100

Throughput \( \lambda \)

©Leonard Kleinrock 2014
Theorem: The optimum value of $N$ which minimizes the mean response time through the switch is: $N=1$
Theorem: The optimum value of $N$ which minimizes the mean response time through the switch is:

$$N = 1$$

### Comparing Architectures

#### Dedicated Resources

<table>
<thead>
<tr>
<th>$\lambda/\mu N$</th>
<th>1</th>
<th>(C/N) bits/sec</th>
<th>2</th>
<th>(C/N) bits/sec</th>
<th>3</th>
<th>(C/N) bits/sec</th>
</tr>
</thead>
</table>

#### Shared Resources

<table>
<thead>
<tr>
<th>$\lambda/\mu$</th>
<th>1</th>
<th>(C/N) bits/sec</th>
<th>2</th>
<th>(C/N) bits/sec</th>
<th>3</th>
<th>(C/N) bits/sec</th>
</tr>
</thead>
</table>

#### LARGE Shared Resources

<table>
<thead>
<tr>
<th>$\lambda/\mu$</th>
<th>1</th>
<th>(C) bits/sec</th>
</tr>
</thead>
</table>

Scale throughput and capacity by a factor of $N$

3. Data Networks

1960’s
Networks of Arbitrary Topology

\[ T = \sum_{i} \frac{\lambda_i}{\gamma} T_i \]

- \( T \) = Average network delay
- \( \lambda_i \) = Traffic on channel i (Msg/sec)
- \( \gamma \) = Network throughput (Msg/sec)
- \( T_i \) = Average delay for channel i

Key equation for network delay.

And it is EXACT!!

Proof

\[ \gamma T = \bar{N} \quad \text{Little’s Result for the full network} \]

\[ \bar{N} = \sum \bar{N}_i \]

\[ \lambda_i T_i = \bar{N}_i \quad \text{Little’s Result for each channel} \]

\[ \gamma T = \sum \lambda_i T_i \]

\[ T = \sum \frac{\lambda_i}{\gamma} T_i \]

©Leonard Kleinrock 2014
The Underlying Principles

• Resource Sharing (demand access)
  • Only assign a resource to data that is present
  • Examples are:
    • Message switching
    • Packet switching
    • Polling
    • ATDM

• Economy of Scale in Networks
  • Bigger is better

• Distributed control
  • It is efficient, stable, robust, fault-tolerant and WORKS!
Economy of Scale in Networks: Cost

Locus of Network Designs

Small Net

Large Net

Slope = $/Kbps

That is, build the largest net possible

©Leonard Kleinrock 2014
4. Finite Population Models

Late 1960’s
Finite Population Models

Thinking

Requesting Service

System of Queues

©Leonard Kleinrock 2014
Finite Population Models

- $M$ = Number of jobs (population size)
- $\lambda$ = Rate of job requests/thinking job
- $1/\lambda$ = Average think time per thinking job
- $T$ = Average Response time in “Service Box”
- $\tau = Cycle Time = 1/\lambda + T$

Input rate of jobs to system = $M \times \lambda \times \frac{1/\lambda}{1/\lambda + T}$

- $1/\mu = Avg \ No. \ of \ opns/job \ (1/\mu C \ sec)$

Output rate of jobs from system = $\mu C (1-p_0)$

$$\mu CT = \frac{M}{(1-p_0)} - \frac{\mu C}{\lambda}$$
Finite Population Models

\[ \mu_{CT} = \frac{M}{(1-p_0)} - \frac{\mu_C}{\lambda} \]
Suppose each job takes exactly $1/\lambda$ sec thinking.

Suppose each job needs exactly $1/\mu C$ sec of service.
Deterministic Model

Suppose each job takes exactly $1/\lambda$ sec thinking

Suppose each job needs exactly $1/\mu C$ sec of service

Now add one more job! And another job!
The “Saturation” Point

• Looks like we just filled the system with 6 carefully placed deterministic jobs.

• In general, without interference of jobs, for this deterministic system, we could fit exactly

\[ M^* = \frac{1/\lambda + 1/\mu C}{1/\mu C} = \frac{\lambda + \mu C}{\lambda} \]

• Let’s define this number as the saturation number, \( M^* \)

Thus we can fit \( M^* \) jobs in and they don’t see each other

The first \( M^* \) jobs look just like 1 job

None of these first $M^*$ see each other.

Each job beyond $M^*$ interferes totally.
5. Flow Control and “Power”

1970’s
Flow Control Issues

• **Routing Procedures:**
  • Easy to design
  • Hard to analyze (dynamic)

• **Flow Control:**
  • Hard to design
  • Outrageously difficult to analyze
  • Absolutely essential

• **Guaranteed to GET you!**
Flow Control in Networks

Throughput

Loss

\[ \lambda - \gamma \]
Flow Control in Networks

Output $\gamma$  $\lambda$  Input $\gamma$  $\lambda$  Throughput $\gamma$

$\gamma_0$

IDEAL

DYNAMIC

CONSERVATIVE

FREE-FLOW

DEADLOCK

Input $\lambda$
Response Time

Throughput

Loss

\[ \lambda \rightarrow \gamma \]

\[ \lambda - \gamma \]

\[ \text{Network Cloud} \]

\[ \text{CAPACITY} \]

\[ \gamma_0 \]

\[ T \]
Response Time vs Throughput

Now let’s ask a good question:

Do you want to operate here?
Or here?

\[ \text{Throughput} \]

\[ \text{Response Time} \]

\[ T(\lambda) \]

\[ \gamma(\lambda) \]
Let's define a new metric of performance:

\[
\text{POWER} \triangleq \frac{\text{Throughput}}{\text{Response Time}}
\]

\[
P = \frac{1}{T(\gamma)}
\]

Power is max when the line out of origin has minimum slope.

\[
dT(\gamma)/d\gamma = T(\gamma)/\gamma
\]
We need a new metric of performance:

\[ \text{POWER} = \frac{\text{Throughput}}{\text{Response Time}} \]

\[ P = \frac{\gamma}{T(\gamma)} \]
Response Time vs Throughput

Let’s Dig Deeper on Understanding

For M/M/1 this gives max Power at $N^* = 1$

Why?

Understand Your Own Results

Use Your Intuition

Only 1 customer in the system

Insight:
Just keep the pipe full!

T = Min
Eff = Max

©Leonard Kleinrock 2014
Understand Your Own Results

- Our intuition says put exactly one person in the queueing system
  - This was from “deterministic” reasoning.
- We can’t actually do that in general
- BUT our earlier result said that we should adjust the system to achieve an average of one person in the queueing system, i.e.,

At Max Power
\[ \bar{N}^* = 1 \]
for M/M/1

Further:
At Max Power we get
- ½ maximum thpt
- 2x minimum delay
for M/M/1
Gee, that’s funny!

What can we say for M/G/1?

\( \bar{N}^* = 1 \)

A More General Power Definition

\[ \text{POWER} \triangleq \left( \frac{\text{Throughput}}{\text{Response Time}} \right)^r \]

\[ P = \frac{\gamma^r}{T(\gamma)} \]

At Max Power

\[ N^* = r \]

for M/M/1
6. Packet Radio

1970’s

Lots of great analysis and design, but the technology would not become available for two decades more
Slotted Aloha

CSMA

1-Persistent CSMA

\[ S = \frac{G[1+G+aG(1+G+aG/2)]e^{-G(1+2a)}}{G(1+2a) - (1-e^{-aG}) + (1+aG)e^{-G(1+a)}} \] (1)

Slotted 1-Persistent CSMA

\[ S = \frac{Ge^{-G(1+a)}[1+a-e^{-aG}]}{(1+a)(1-e^{-aG})+ae^{-G(1+a)}} \] (2)

Non-Persistent CSMA

\[ S = \frac{Ge^{-aG}}{G(1+2a)+e^{-aG}} \] (3)

Slotted Non-Persistent CSMA

\[ S = \frac{aGe^{-aG}}{(1+a)(1-e^{-aG})+a} \] (4)

p-Persistent CSMA

\[ S(G, p, a) = \frac{(1-e^{-aG})[P_s'\pi_0+P_s(1-\pi_0)]}{(1-e^{-aG})[a\tilde{\tau}'\pi_0+a\tilde{\tau}(1-\pi_0)+1+a]+a\pi_0} \] (5)

CSMA

1-Persistent CSMA

\[ S = \frac{G[1+G+aG(1+G+aG/2)]e^{-G(1+2\alpha)}}{G(1+2\alpha)-(1-e^{-aG})+(1+aG)e^{-G(1+a)}} \] (1)

Slotted 1-Persistent CSMA

\[ S = \frac{Ge^{-G(1+a)}[1+a-e^{-aG}]}{(1+a)(1-e^{-aG})+ae^{-G(1+a)}} \] (2)

Non-Persistent CSMA

\[ S = \frac{Ge^{-aG}}{G(1+2\alpha)+e^{-aG}} \]

Slotted Non-Persistent CSMA

On the airplane home

Plus

• Hidden Terminals
• Busy Tone
• Reservation

Distributed Multi-Access

- Performance degradation from pure queueing due to:
  - Unpredictable arrival times
  - Unpredictable service times
- We also lose performance because we do not know who is on queue in a distributed environment
## The Price for Forming the Queue

<table>
<thead>
<tr>
<th>Control Type</th>
<th>Collisions</th>
<th>Idle Capacity</th>
<th>Control Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Control (e.g. Aloha)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Static Control (e.g. FDMA)</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Dynamic Control (e.g. Reservation)</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Giant Stepping in Packet Radio
Giant Stepping in Packet Radio

- Multihop
- Each hop covers distance $R$ (Tx Radius)
- Total distance to cover is $D$ ($D \gg R$)
- Big $R$, more interference, fewer hops
- Small $R$, less interference, more hops
- $T(R)$ is mean response time per hop
- $T = \text{Total Delay} = T(R)[D/R]$
- Choose $R=R^*$ to minimize total delay
- $dT(R)/dR = T(R)/R$ optimality condition
\[ \frac{dT(R)}{dR} = \frac{T}{R} \]

Optimum Radius \( R^* \)

7. A Generalization
This is the 3\textsuperscript{rd} Time We Have Seen This Today!

Is there a General Case Here?

\[ P = \frac{\gamma}{T(\gamma)} \]

\[ \text{Giant Stepping} \quad \text{Response Time} \quad \frac{dT(R)}{dR} = \frac{T}{R} \]

\[ \text{Optimum Radius } R^* \]

\[ \text{Locus of Network Designs} \]

\[ \text{Economy of Scale} \]

\[ \text{Small Net} \quad \text{Large Net} \]

\[ \text{Slope} = \text{Kbps/\$} \]
The General Case

Maximize $\frac{\text{Good}}{\text{Bad}} = \text{Minimum slope line}$

Slope = $\frac{\text{Bad}}{\text{Good}}$

So operate at point where line out of origin has minimum slope

Kleinrock, L., "Optimizing the Ratio of Good/Bad" in preparation
8. Distributed Processing

1980’s
The General Series/Parallel Processing Net

\[ \lambda = \sum_{k=1}^{m} \lambda_k \]

\[ C = \sum_{k=1}^{m} C_k n_k \]

The Pure Single Node

\[ T_0 = \frac{1}{\mu C - \lambda} \quad \text{M/M/1} \]

\[ \frac{1}{\mu} = \text{Avg No. of opns/job} \]

The General Series/Parallel

\[ T = \sum_{k=1}^{m} \frac{\lambda_k}{\lambda} n_k T^{(k)} \]

\[ T^{(k)} = \frac{1}{\mu n_k c_k - \lambda_k} \]
Ratio of General/Single Node

\[
\frac{T}{T_0} = \sum_{k=1}^{m} n_k \frac{\rho_k / (1 - \rho_k)}{\rho / (1 - \rho)}
\]

\[
\rho_k = \frac{\lambda_k}{\mu n_k c_k}
\]

\[
\rho = \frac{\lambda}{\mu c}
\]

Let’s look at some special cases:
The Pure Tandem

- $m=1, \ n_1=n, \ \lambda_1 = \lambda, \ C_1 = C/n$

$$\frac{T}{T_0} = n$$
The Pure Parallel System

- $n_k = 1, \quad \lambda_k = \lambda/m, \quad C_k = C/m \quad \text{for } k=1,2,\ldots,m$

\[ T = \frac{T_0}{m} \]
The Symmetric Series-Parallel System

- $n_k = n$, $\lambda_k = \lambda/m$, $C_k = C/mn$ for $k = 1, 2, \ldots, m$

\[ T = \frac{T_0}{mn} \]
The General Series/Parallel System with Uniform Traffic

\[ \lambda_k = \frac{\lambda}{m} \]

\[ \frac{T}{T_0} = \sum_{k=1}^{m} n_k \]

Bigger and fewer is better
9. Latency/Bandwidth Tradeoff

1990’s
The Latency/Bandwidth Tradeoff

from Kilobits to Megabits to Gigabits!

Evolution, Revolution or Bump?
How Fast is a Gigabit?

• A billion bits/sec is really fast!

• But ... the speed of light isn’t!
One Megabit File

780

64 Kbit/sec

©Leonard Kleinrock 2014
One Megabit File

1.5 Megabit/sec
We seem to have bumped into the speed of light!

or

Something’s going “bump” in the light!
When Did We Hit the Bump?

At some CRITICAL capacity!

Response Time = Queueing + Tx Time + Latency

Define Critical Capacity to be the point where:

Queueing + Tx Time = Latency

Kleinrock, L., "The Latency/Bandwidth Tradeoff in Gigabit Networks", IEEE Communications Magazine, April 1992, Vol.30, No.4, pp.36-40
The Latency-Bandwidth Tradeoff

- Queueing + Tx Time = Latency
- C < Critical → Bandwidth Limited
- C > Critical → Latency Limited
Critical Bandwidth

Queueing + Tx Time = Latency

- 100 GBPS
- 10 GBPS
- 1 GBPS
- 100 MBPS
- 10 MBPS
- 1 MBPS
- 100 KBPS
- 10 KBPS

Latency Limited

Bandwidth Limited

(1 MEGABIT FILES) (CROSS COUNTRY)

©Leonard Kleinrock 2014
20 Million Bits in the pipe!

AT 1 GBPS
**Key System Parameter**

L = Cable Length (kilometers)
PD = 5L (microseconds)
C = Bandwidth (megabits/sec)
b = Packet Length (bits)

\[
a = \text{Propag Delay/Pkt Tx Time} = \frac{5LC}{b} \text{ (# packets in cable)}
\]

<table>
<thead>
<tr>
<th></th>
<th>SPEED MBPS</th>
<th>PKT LNGTH BITS</th>
<th>PROP DELAY MICROSEC</th>
<th>LATENCY a</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIRELESS NET</td>
<td>10.0</td>
<td>1,000</td>
<td>5</td>
<td>.05</td>
</tr>
<tr>
<td>1 kilometer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOCAL NET</td>
<td>1,000.00</td>
<td>1,000</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1 kilometer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIBER LINK</td>
<td>1,000.00</td>
<td>1,000</td>
<td>20,000</td>
<td>20,000</td>
</tr>
<tr>
<td>Cross country</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

©Leonard Kleinrock 2014
The Latency-Bandwidth Tradeoff

\[
C_{\text{crit}} = \frac{b}{5L(1-\rho)}
\]

or

\[
a = \frac{1}{1-\rho}
\]

where

\[
\rho = \text{Load} = \frac{\lambda b}{C}
\]

\[
C \text{ (Mbps)}, \quad b \text{ (bits/msg)}
\]

\[
L \text{ (Km)}, \quad \lambda \text{ (msg/microsec)}
\]

\[
a = \text{Propag Delay/Pkt Tx Time} = \frac{5LC}{b} \text{ (# packets in cable)}
\]

Kleinrock, L., "The Latency/Bandwidth Tradeoff In Gigabit Networks", IEEE Communications Magazine, April 1992, Vol.30, No.4, pp.36-40
Latency vs Bandwidth

Bandwidth (C)

Latency (L)

Packet Length (b)

Cable Length (L)

a = $5 \frac{LC}{b}$

USA

LC = $4 \times 10^6$

20 Mbit

LC = $10^7$

LC = $10^6$

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit

100 Mbit

1 Gbps

100 Mbps

1 Mbit

10 Mbit
Latency vs Bandwidth

Latency

Bandwidth

USA

10,000 Km

1,000 Km

100 Km

LC=10^5

LC=10^6

LC=10^7

100 Mbit

10 Mbit

1 Mbit

100 Mbps

1 Gbps

10 Gbps

a = \frac{1}{1 - \rho}

a = 5 \text{ LC/b}
Gigabit Networking
Fundamental Issues

• **Speed of Light is Too Slow:**
  • 20,000 Microsec to cross USA
  • 20 Million bits in a Gigabit pipe
  • Control signals suffer enormous delays

• **Global Information is Costly:**
  • It takes:
    bandwidth, time, processing, storage.
  • It will be:
    delayed, stale, wrong, incomplete.

©Leonard Kleinrock 2014
10. The Gur Intelligent Agent
1990’s
Adaptive Agents and The Gur Algorithm

1. Each Agent votes YES or NO
2. A fraction $f$ votes YES
3. Using a function $p(f)$ which is unknown to them, a referee gives (takes) $1$ from each independently with probability $p$
4. Go to step 1 and repeat!
Can We Construct The Players to Seek the Optimum Behavior?

Yes!
How Is It Done?

Design each player as a finite-state discrete-time automaton with 2N states.

Vote NO

Reward => Edge seeking behavior

Punishment => Center seeking behavior

Vote YES

11. Optimal Update Times

2000’s
Optimal Update Times for Out-of-Date Information

• Problem:
  When and how often should a user update a given piece of information as it goes further and further out-of-date?

• Assumptions:
  There is a cost $C > 0$ of updating a given piece of information.
  There is an expected value per unit time associated with having a piece of information that was updated $t$ time units ago.
  • This value is $f(t)$.

• Question:
  Given $f(t)$ and $C$, when and how often should a user update a given piece of information?
Value of Out-of-Date Information $f(t)$

Average Value Gained per Unit Time is **maximum** when

$$\int_{t=0}^{x} f(t)\,dt - C = f(x)$$

Why?
Value Gained Over Multiple Updates

The diagram illustrates a function $f(t)$ over time $t$. The x-axis represents time intervals marked as $x$, $2x$, $3x$, and $4x$, with corresponding function values at these points. The function $f(t)$ shows a periodic pattern, indicating the value gained over multiple updates.
12. Peer-to-Peer File Systems

2000’s
Peer-to-Peer File Networks

- Distributed file sharing network
- The service consumers are the service providers as well
- Files uniformly distributed in net
- Search using controlled flooding
- How many copies of a file should be stored?
Definitions

- $M = \text{number of nodes in the system}$
- $N = \text{number of unique files in the system}$
- $K = \text{per-node storage size in number of files}$
- $\lambda_i = \text{request rate for file } i \text{ per node}$
- $\lambda = \sum_{i=1}^{N} \lambda_i = \text{total input rate per node}$
- $n_i = \text{number of replicas of file } i \text{ in the system}$
- How should select $n_i$?
Minimum Search Distance

\[ \tau_i (n_i) = \text{Average shortest distance from a querying node to a replica of file } i \]

\[ \tau_i (n_i) = \alpha \log \frac{M}{n_i} \]

\[ \tau = \text{Avg search distance} = \sum_{i=1}^{N} \frac{\lambda_i}{\lambda} \tau_i (n_i) \]

**Minimize** \( \tau \) \{\( n_i \)\}

\[ n_i = \lambda_i \frac{\text{KM}}{\lambda} \]

©Leonard Kleinrock 2014
Further Results

Why shouldn’t I store only unpopular files?

Given

\[ n_i = \frac{\lambda_i KM}{\lambda} \]

- Each replica of file \( i \) serves

\[ M \frac{\lambda_i}{n_i} = \frac{\lambda}{K} \text{ requests/sec} \]

- Each node has \( K \) files, so the load on each node is \( \lambda \) requests/sec. **Don’t play games.**
- So each node has exactly the same load!
- If queueing delays are convex in node utilization, the average download time is **minimized.**

13. Guidelines for Research
My Five Golden Guidelines to Research

1. Conduct the 100-year test.
2. Don’t fall in love with your model.
4. Understand your own results.
5. Look for “Gee, that’s funny!”
"Why do so few scientists make significant contributions and so many are forgotten in the long run?"

“If you don't work on important problems, it's not likely that you'll do important work.”

1. The 100 Year Test

• Hamming once asked me,

“What progress of today will be remembered 1000 years from now?”

Let’s simplify it: Will your work be remembered 100 years from today?
2. But Don’t Fall in Love With Your Model

The Real World → Mathematical Model of The Real World → Solution to the Mathematical Model

Approximation
3. Beware of Mindless Simulation
Ask the Obvious Questions
4. Understand Your Own Results

• Take the time to think deeply about your results.
• Use deterministic or simple models to explain behavior
  • e.g. why does “filling the pipe” make sense
• Think about upper and lower bounds
• Take limits to force behavior
• Look at extreme cases to check validity and intuition
5. Look for “Gee, that’s funny!”

• Don’t ignore strange looking results
  • Often that’s where the “gold” lies

• The greatest scientific discoveries are Not accompanied by “Eureka”, but most occur when someone mutters, “That’s interesting”
More on Modeling

• Moving the frontier is tough (we mislead our students)
• Once they move it, they will be able to repeat it again (students don’t believe us)
• Teach your students to understand their results!
• Generalization usually comes when you can see the simplicity of a solution
• As Norbert Wiener said, “Every scientist must occasionally turn around and ask not merely “How can I solve this problem?” but, “Now that I have come to a result, what (other) problems have I solved?”
• When a field gets too crowded, move your research vector slightly
• Keep your interest in related areas, areas where something might happen.
Thank You

www.lk.cs.ucla.edu