Work per Task: a Useful Measure for Increasing Throughput While saving Energy

Yitzhak (Tsahi) Birk

Technion

www.ee.technion.ac.il/~birk

www.psl.technion.ac.il

Energy-Performance Tradeoff

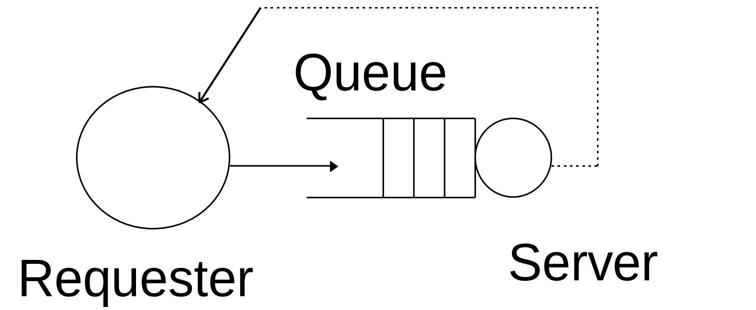
- **“Performance” and energy conservation are often conflicting goals.**
  - **A faster CPU is often less energy-efficient**
  - **Driving faster consumes more energy for the trip**
  - **A disk that seeks faster consumes more energy per seek**
- **Is this always the case?**
- **Can we do anything?**

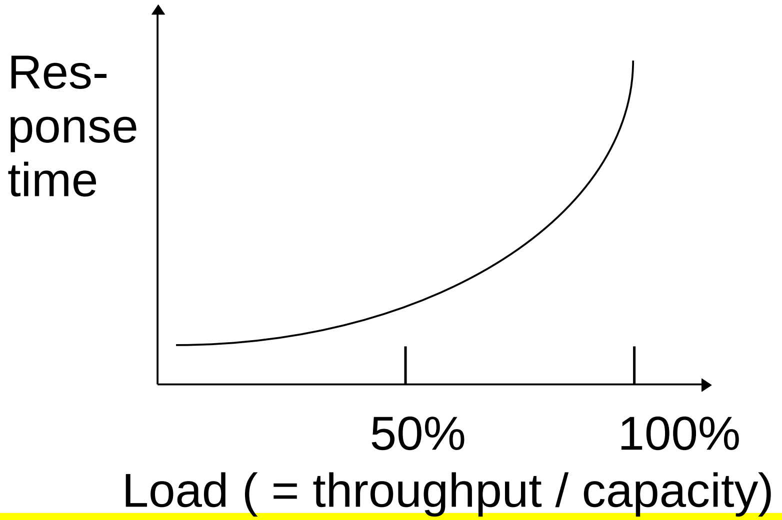
Outline

- **Performance measures and the relationship among them**
- **“Work per task”**
- **Energy-performance win-win situations**
- **Conclusions**

Performance Measures

- **User perspective: time to accomplish mission (delay; response time; latency)**
  - **queuing delay**
  - **service time**





- **Service-provider perspective: throughput (Missions/sec; bits/sec; transcations/sec)**

Is service time reduction the common goal?

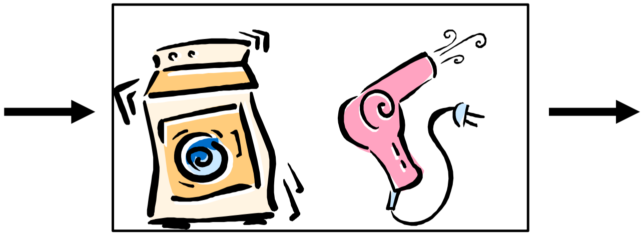
Service Time = 1/throughput?

Pipelining

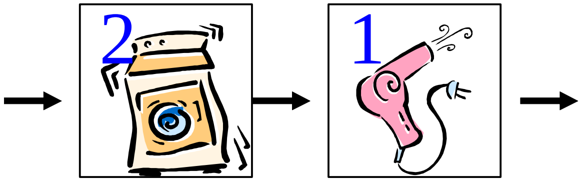
Tasks:

- **Wash: 55min**
- **Dry: 45 min**
- **(Transfer: 5 min)**

5+100+5



5+55+5+45+5



	Maximum throughput	Service time
Combo	1/110	110
Separate	1/65	115

Increases both throughput and service time!

Work Per Task

- **Unit: seconds (time), but**
- **Unlike latency, this is actually (resource x time), like “person hours”**
- **In time interval T<sec>, an N-resource system can do N·T<sec> of work**
- **If a task requires less work, more tasks can be done during time T**

Less work per task ⇒ higher throughput

Work Per Task and Energy

- **If a resource consumes fixed power, reducing the amount of work (as defined!) per given task reduces energy consumption for a given workload.**

Less work per task ⇒ potentially save energy!

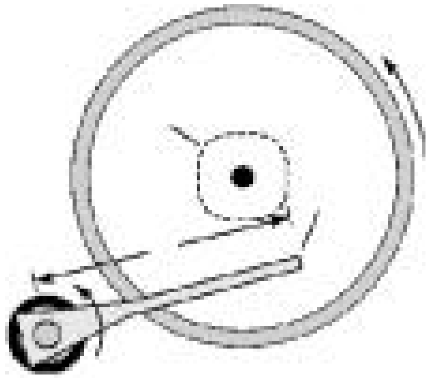
Reducing work per task may lead to win-win for energy and performance!

Example 1

Data Layout for Random Block-Set Access

Scenario

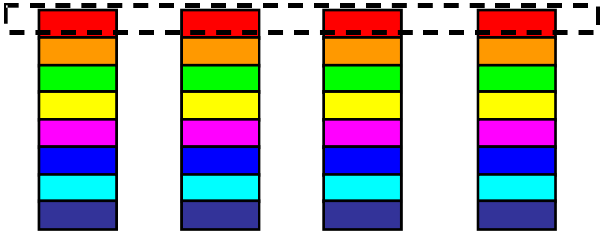
- **Data: numerous disjoint sets of 4 blocks.**
- **Workload: numerous requests to read entire sets, chosen randomly.**



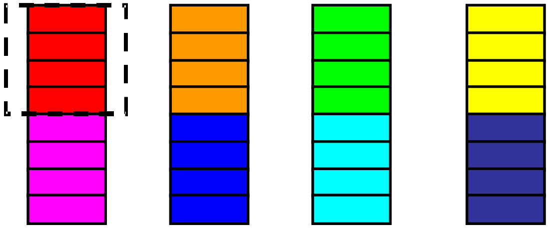
- **Hardware: 4 disk drives**
  - **Time to access a block: t_s**
  - **Time to transfer a block: t_t**
- **Controller can immediately assign a request to an idle disk (all disks are kept busy at all times).**
- **Goal: maximize throughput (block-sets / sec).**

Data Layout Options

- **Striping each block set across all disks**



- **Each block set on a single disk**

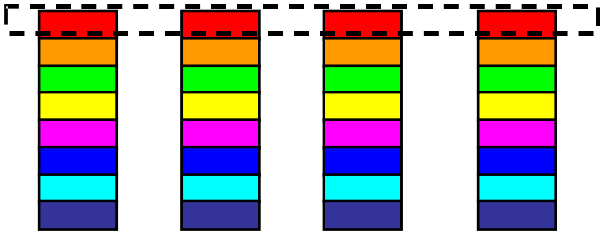


- **For both options: assume perfect load balancing and that all disks are busy all the time.**

Work Per Task (reading a block set)

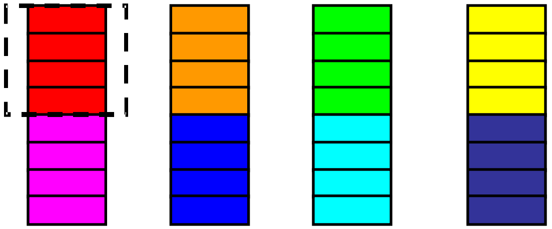
- **With Striping:**

$$W = 4(t_s + t_t)$$



- **No Striping:**

$$W = t_s + 4 \cdot t_t$$

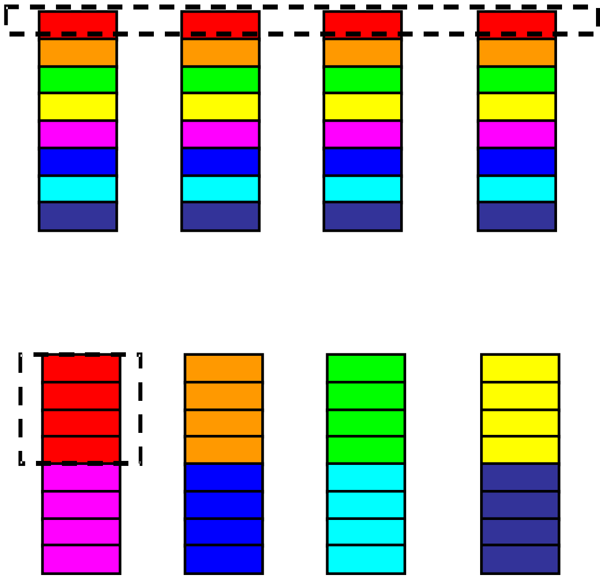


The unstriped layout is more efficient

Throughput Comparison

- S_{max}= 4/W,

  where W is the amount of disk work per task

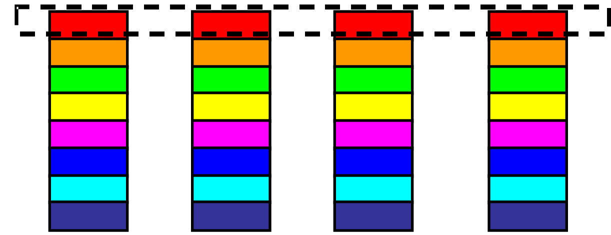


$$\frac{S_{unstriped}}{S_{striped}} = \frac{4\left(T_s + T_t\right)}{T_s + 4T_t}$$

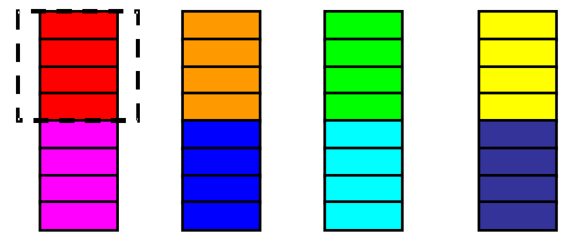
The non-striped layout wins on throughput

Energy Comparison

- **Seek**:
  - Same energy per seek
  - Striped: 4 seeks per task
  - Non-Striped: 1 seek per task
- **Rotation**:
  - Fixed power ⇒ ∝ W
- **Signal processing**: same
- **Communication protocols**:
  less with no striping



$$W = 4 \cdot t_s + 4 \cdot t_t$$



$$W = t_s + 4 \cdot t_t$$

The non-striped layout also wins on Energy

What About Response Time?

- Wasn't part of the requirement!, but let's examine it anyhow.

- "Zero load":
  - Small blocks: ~equal
  - Large blocks: transfer time dominates, so stripe the data at negligible throughput and energy penalty.

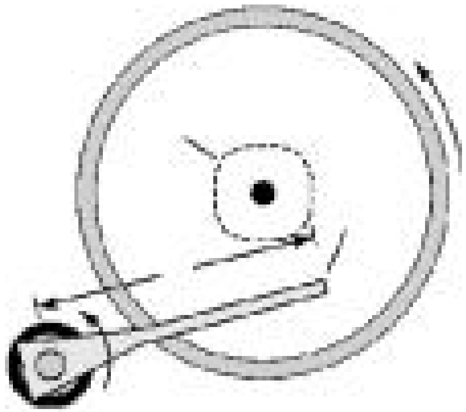
- Heavy load:
  - Response time is dominated by queuing delay
  - Higher maximum throughput $\Rightarrow$ shorter queues for any given workload $\Rightarrow$ non-striped wins!

Example 2

Web Server

Background

- **Web server:**
  - Stores data on disk
  - Retrieves it in response to requests

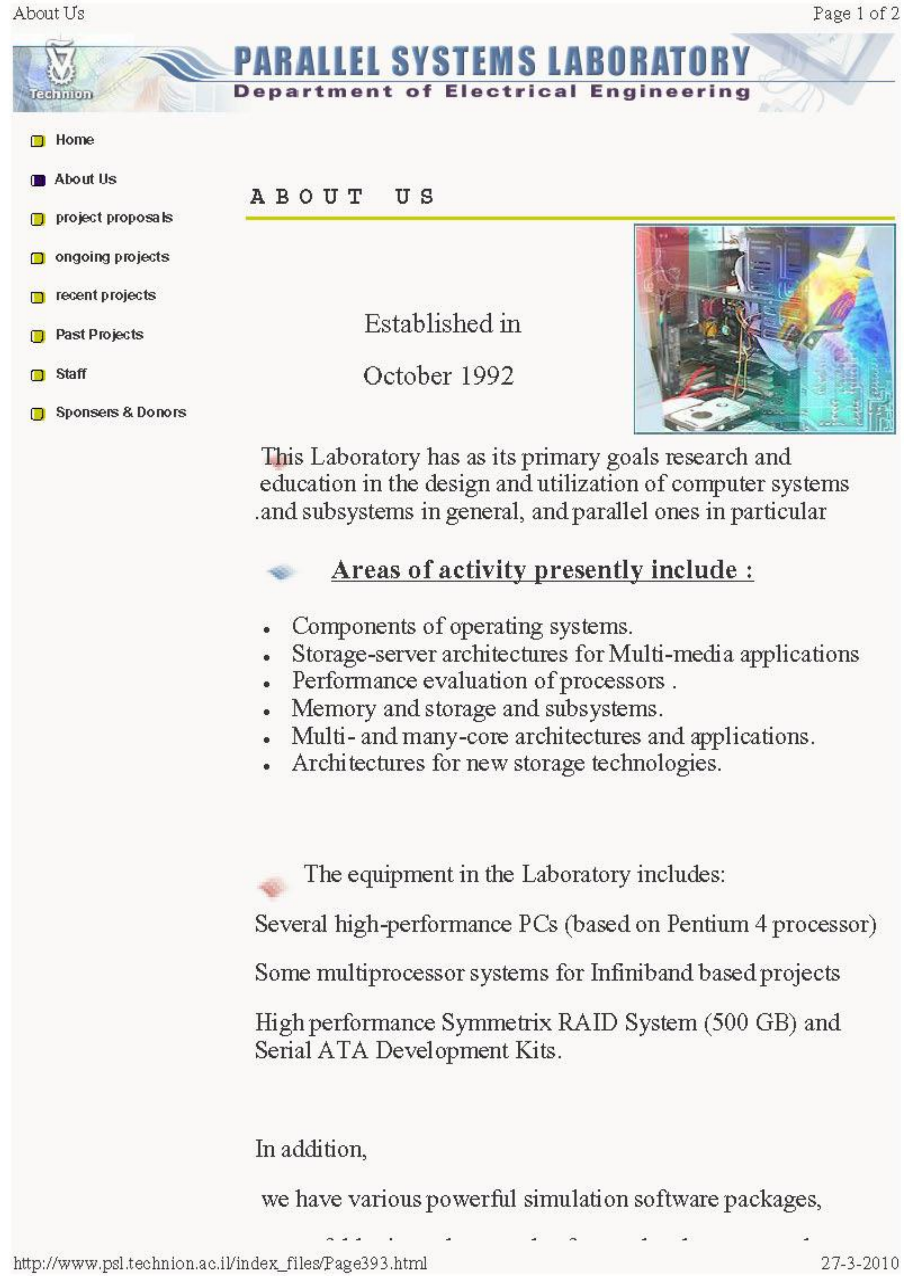


- **Web page:**
  - HTML “skeleton”
  - Multiple small embedded objects
- **Each page requires multiple disk accesses:**
  - Wasted disk time → low maximum throughput
  - Extra disk seeks and time → more energy per workload
- **Goal: reduce work per page**

PSL Home Page

www.psl.technion.ac.il

- **Total size: ~150kB**
- **Composition:**
  - Skeleton
  - 8 embedded objects
- **Disk work:**
  - ~80ms
  - 9 seeks
- **Max disk throughput: ~ 12 pages/sec**


About Us Page 1 of 2

PARALLEL SYSTEMS LABORATORY
Department of Electrical Engineering

- Home
- About Us
- project proposals
- ongoing projects
- recent projects
- Past Projects
- Staff
- Sponsers & Donors

ABOUT US

Established in

October 1992

This Laboratory has as its primary goals research and education in the design and utilization of computer systems .and subsystems in general, and parallel ones in particular

Areas of activity presently include :

- Components of operating systems.
- Storage-server architectures for Multi-media applications
- Performance evaluation of processors .
- Memory and storage and subsystems.
- Multi- and many-core architectures and applications.
- Architectures for new storage technologies.

The equipment in the Laboratory includes:

Several high-performance PCs (based on Pentium 4 processor)

Some multiprocessor systems for Infiniband based projects

High performance Symmetrix RAID System (500 GB) and Serial ATA Development Kits.

In addition,

we have various powerful simulation software packages,

http://www.psl.technion.ac.il/index_files/Page393.html 27-3-2010

Possible Approaches

- **Prefetching**:
  - Mostly latency hiding, not a reduction in disk work per web page
  - (Smart scheduling can reduce mean seek distance and thus some reduction in work)
- **Contiguous placement of page's objects**:
  - Rotational latency not saved
  - Seek may not be saved if server is busy, as other requests are interlaced with those of page
  - Not effective unless also read together

Packaging

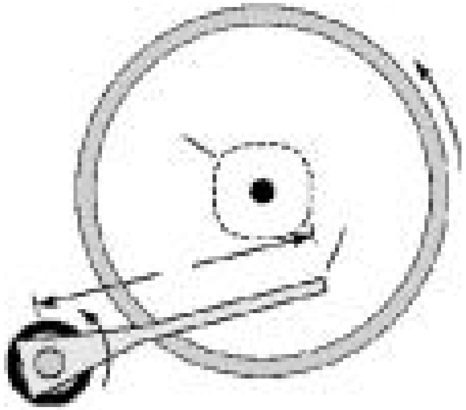
- **All page's objects:**
  - **placed contiguously on disk**
  - **read in a single disk access.**
- **After reading from disk:**
  - **Server separates in memory (unaware client), or**
  - **Sends entire package to client (participating client).**
- **Price: replication of objects that are part of multiple pages (negligible or don't do it)**
- **Complication: need to update upon object change**
  - **Server uses "check if modified since" after reading package**
  - **Various policies are possible upon change, ranging from "do nothing" to look for all copies and update them.**

Packaging: Impact (PSL example)

- 9 seeks → 1 seek per page
- Disk work: 80ms → 10ms
- Disk throughput: 12 pages/sec → 100 pages/sec
- Disk energy per web page:
  - Seek: down 9X
  - Motor: down ~9X (same power but for a shorter time)
  - Electronics: some savings

Packaging increases throughput & saves energy!

Ex 3: Same Work for Less Energy



- **Disk access:**
  - Seek
  - Rotational Latency
  - Transfer
- **Important insight:**
  - Slower seek need not prolong the time to first bit
  - Slower seek consumes less energy
  - Seek speed can be chosen per seek (unlike rpm)

- **Implication: can adjust seek speed based on seek distance and relative rotational position so as to save energy with no performance penalty! [Toshiba]**

Ex 4: Scheduling for Efficient Access

- Reduces work and increases throughput.

- Fairness:
  - Avoid starvation by gating
  - At heavy load,
    higher max throughput $\Rightarrow$ lower load
    $\Rightarrow$ shorter queue
    $\Rightarrow$ shorter response times for all!

Caution: Power, Work and Energy

- **Energy = Work · Power,**

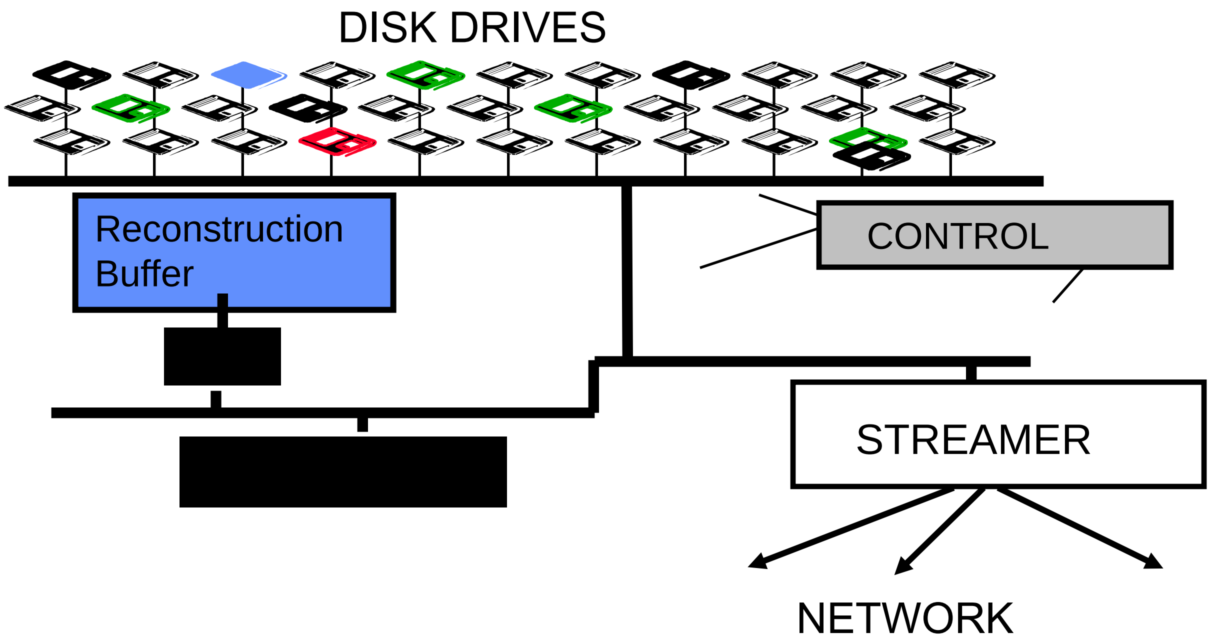
  so

- **Reducing Work by increasing Power may not reduce Energy consumption**

Example 5

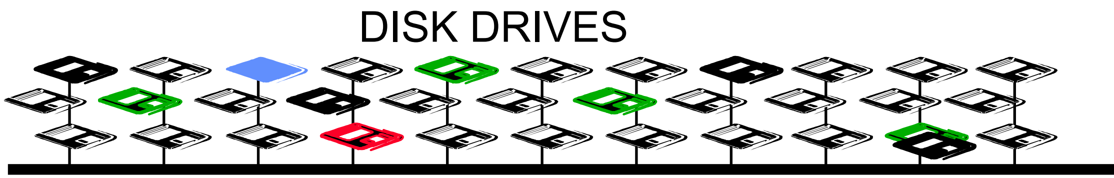
RandomRAID Video Server

Random RAID Video Server [1995]



RandomRAID - Overview

- **N disk drives**



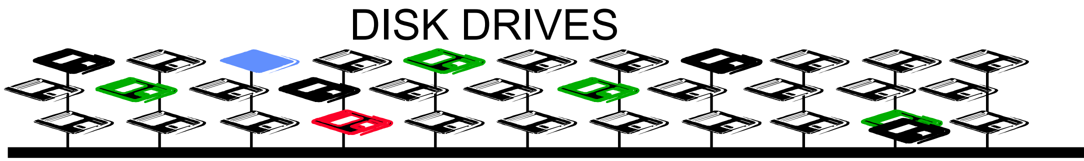
- **Parity groups:**
  - **Stripe size k: arbitrary k < N**
  - **Choice of disk drives constituting a parity group (at write time): “random”**

- **Full-stripe reads (e.g., data streaming):**
  - **pick any (k-1) of the k relevant disks**
  - **Example: the ones with the shortest queues or the ones that will do minimum work**

- **Special case: random mirroring**

RandomRAID – Salient Features

- **Load balancing**
  - **During normal operation**
  - **In degraded mode**
  - **During reconstruction**



- **Work reduction or shorter response time (depends on disk-choice policy)**
- **Incremental scale-up possible, including non-identical disk drives**
- **Offers the benefits of randomization without the negative side-effects!**

Judicious exploitation of redundancy for performance enhancement and energy efficiency!

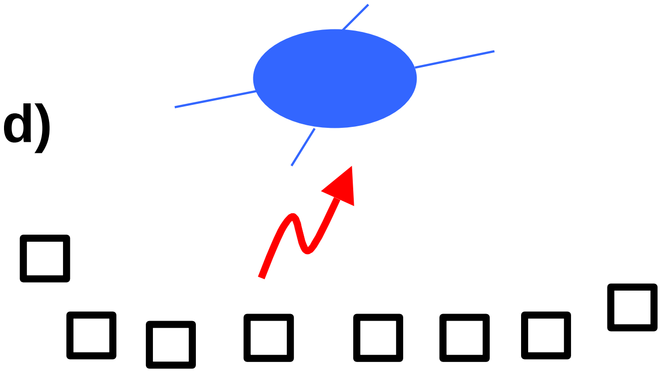
Example 6

On-Line Transaction Processing

via Satellite

(e.g., cash register)

Maximizing Deadline-Constrained Capacity in Multi-channel ALOHA Networks [Birk et al]

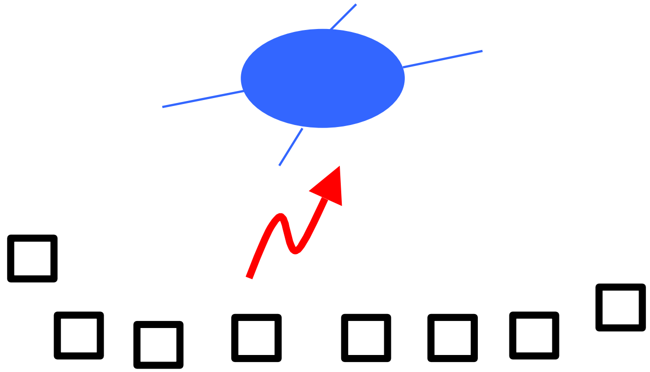


- **Multichannel ALOHA:**
  - Upstream contention channels (shared)
  - Private downstream channel for hub transmissions and Acks
  - New message: draw a channel randomly, transmit and wait for Ack
  - If no Ack, redraw channel and retransmit

- **Transaction processing – performance goals:**
  - User: deadline and permissible Pr(failure)
  - Service provider's goal: max. transactions/sec

- **For battery-operated terminals and sensors: minimum mean energy per transaction.**

Multi-Channel ALOHA: Example

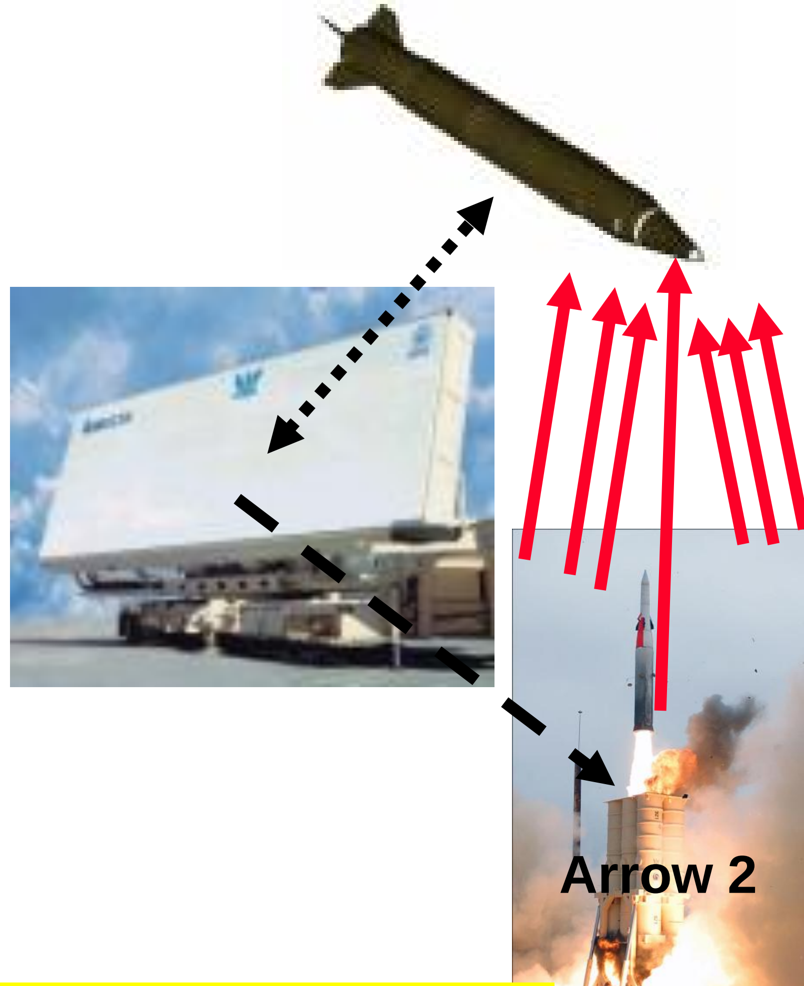


- **Scenario:**
  - **Delay permits 2 attempts (rounds).**
  - **P(collision) = 0.1**
  - **P(failure) = 0.001**
- **Greedy approach: send 3 copies in 1st attempt:**
  - **minimum mean delay**
  - **3 copies per message**
- **Better approach:**
  - **Send 1 in first attempt**
  - **Send 2 in 2nd attempt iff 1st fails**
- **Analysis**
  - **longer mean delay (who cares?)**
  - **1.2 copies per message**
  - **2.5-fold higher capacity**
  - **2.5x less energy per transaction!**

Higher Throughput and Less Energy!

The Arrow ABM System [Dov Raviv]

- **Arrow miss probability: 0.1**
- **System requirement: 0.001**
- **Possible solution: 3 arrows**
- **Key observations:**
  - **Advance radar warning allows a 2nd attempt upon failure.**
  - **Interception time is not important.**
- **Optimal solution:**
  - **Fire one arrow**
  - **Iff it misses, fire two more**
  - **Probability of failure: 0.001**
  - **Average of 1.2 Arrows per target**



Exact requirement + smart solution ⇒ 60% savings!

Learn from other applications!

Conclusions

- While energy reduction may be at odds with other goals, reduction of the amount of work per task (not service time!) can create win-win situations.
- Redundancy can be beneficially exploited for performance enhancement, not only for fault tolerance.
- Designing to the true requirements may lead to interesting opportunities.