CSC2231: Coding + Bloom Filters + Tiger

http://www.cs.toronto.edu/~stefan/courses/csc2231/05au

Stefan Saroiu

Department of Computer Science

University of Toronto

Administrivia

- **Schedule for the rest of the semester:**
  - Wednesday: project presentations (from 4 to 6:30pm)
  - Thursday: wrap-up lecture/introspection <-- don't miss this
  - Thursday: project write-up due
  - Following 2 weeks after that: food + sleep!
- **Have you started working on your slides?**
- **8 slides only, 30 minute slot. Shoot for 20 mins presentation**

Outline

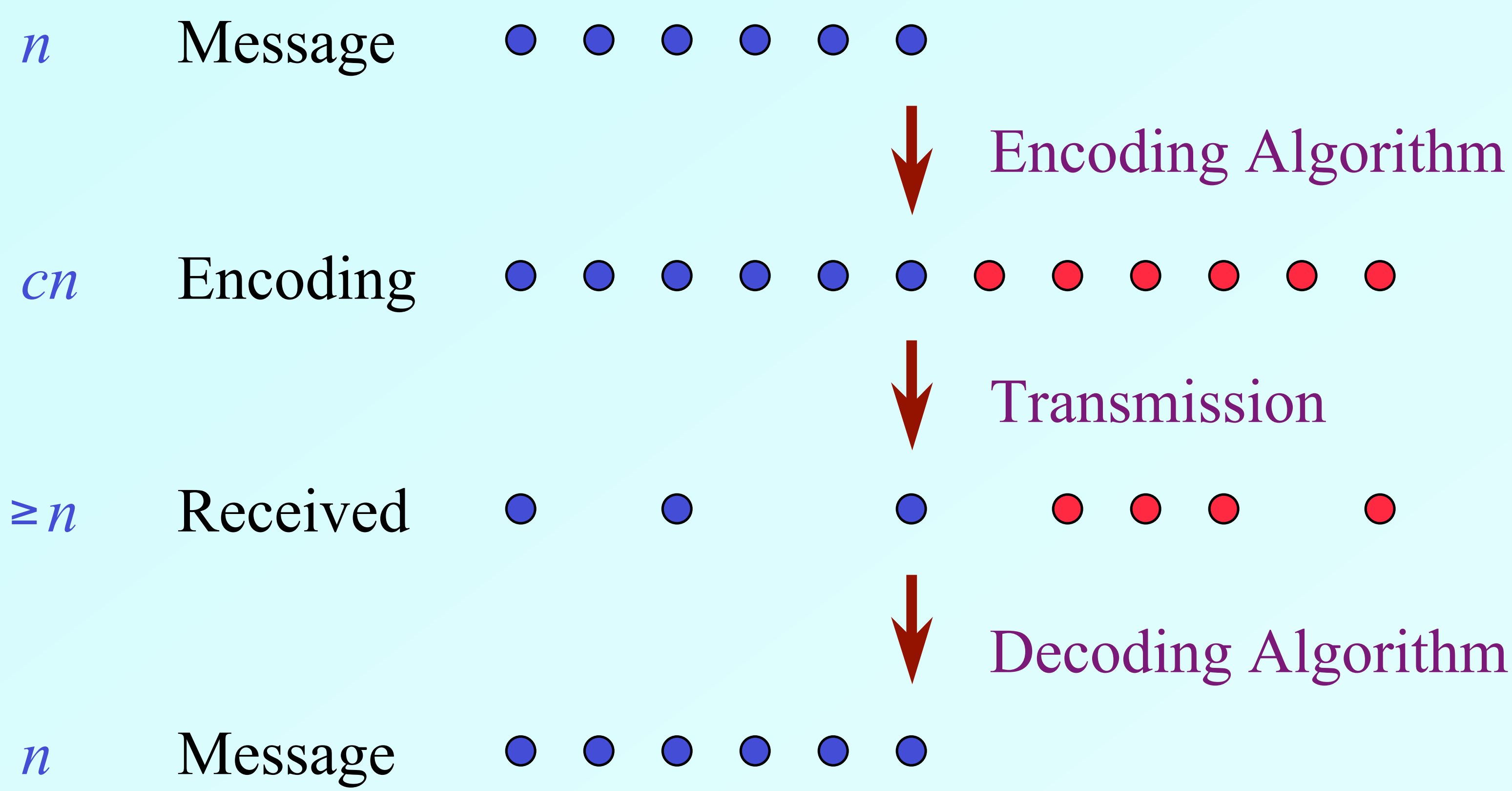
- **Erasure codes**
  - Digital Fountain
- **Bloom Filters**
  - Summary Cache, Compressed Bloom Filters
- **Tiger**

- **Many of today's slides are from Michael Mitzenmacher's talks!**

Codes: High Level Idea

- **Everyone thinks of data as an ordered stream.** ***I need packets 1-1,000.***
- **Using codes, data is like water:**
  - You don't care what drops you get.
  - You don't care if some spills.
  - You just want enough to get through the pipe.
  - *I need 1,000 packets.*

Erasure Codes



Application: Trailer Distribution Problem

- **Millions of users want to download a new movie trailer.**
- **32 megabyte file, at 56 Kbits/second.**
- **Download takes around 75 minutes at full speed.**

Point-to-Point Solution Features

Point-to-Point Solution Features

- **Good**
  - Users can initiate the download at their discretion.
  - Users can continue download seamlessly after temporary interruption.
  - Moderate packet loss is not a problem.
- **Bad**
  - High server load.
  - High network load.
  - Doesn't scale well (without more resources).

Broadcast Solution Features

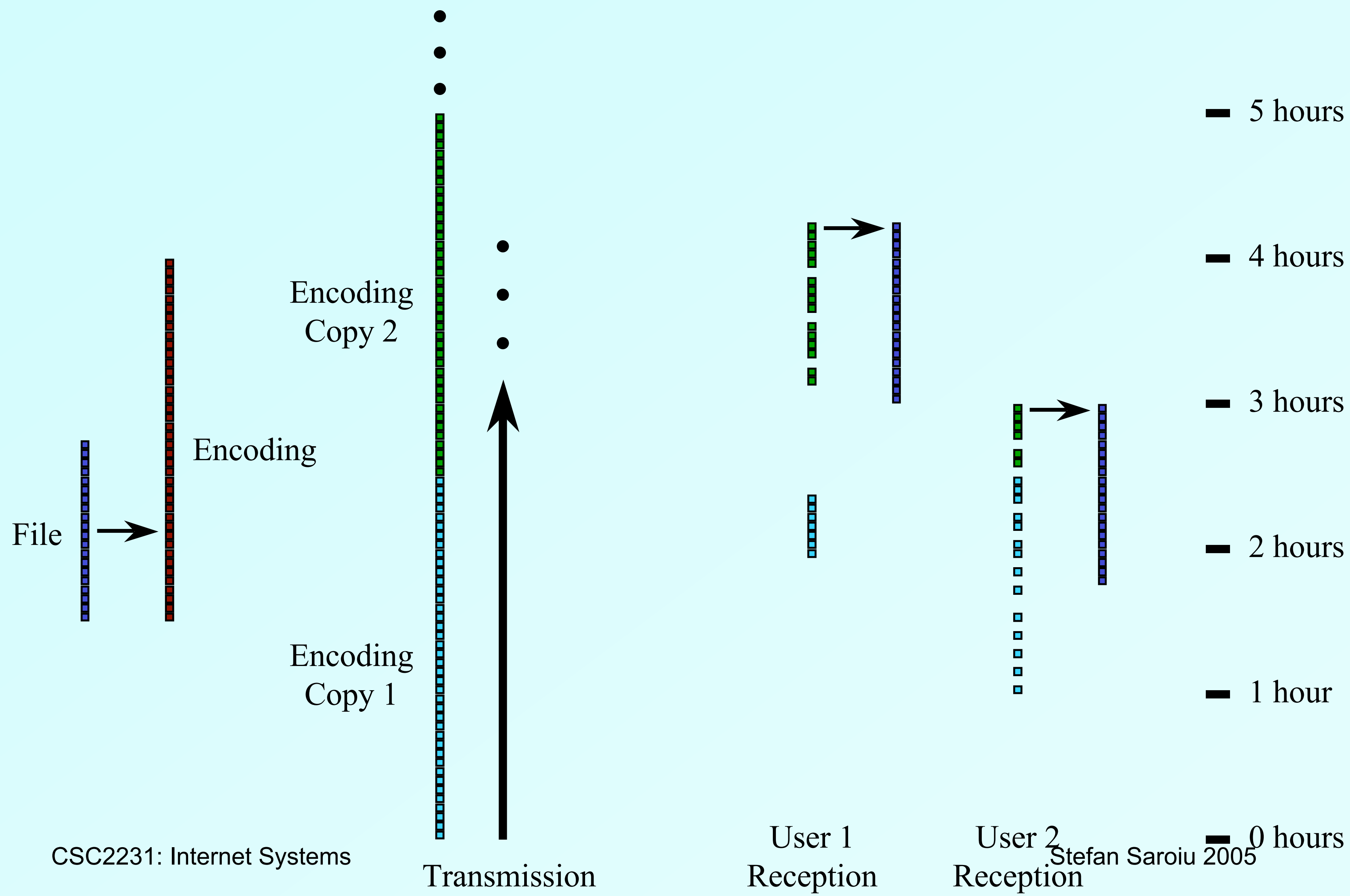
Broadcast Solution Features

- Bad
  - Users cannot initiate the download at their discretion.
  - Users cannot continue download seamlessly after temporary interruption.
  - Packet loss is a problem.
- Good
  - Low server load.
  - Low network load.
  - Does scale well.

A Coding Solution: Assumptions

- **We can take a file of *n* packets, and encode it into *cn* encoded packets.**
- **From any set of *n* encoded packets, the original message can be decoded.**

Coding Solution



Coding Solution Features

- Users can initiate the download at their discretion.
- Users can continue download seamlessly after temporary interruption.
- Moderate packet loss is not a problem.
- Low server load - simple protocol.
- Does scale well.
- Low network load.

So, Why Aren't We Using This...

- **Encoding and decoding are slow for large files -- especially decoding.**
- **So we need fast codes to use a coding scheme.**
- **We may have to give something up for fast codes...**

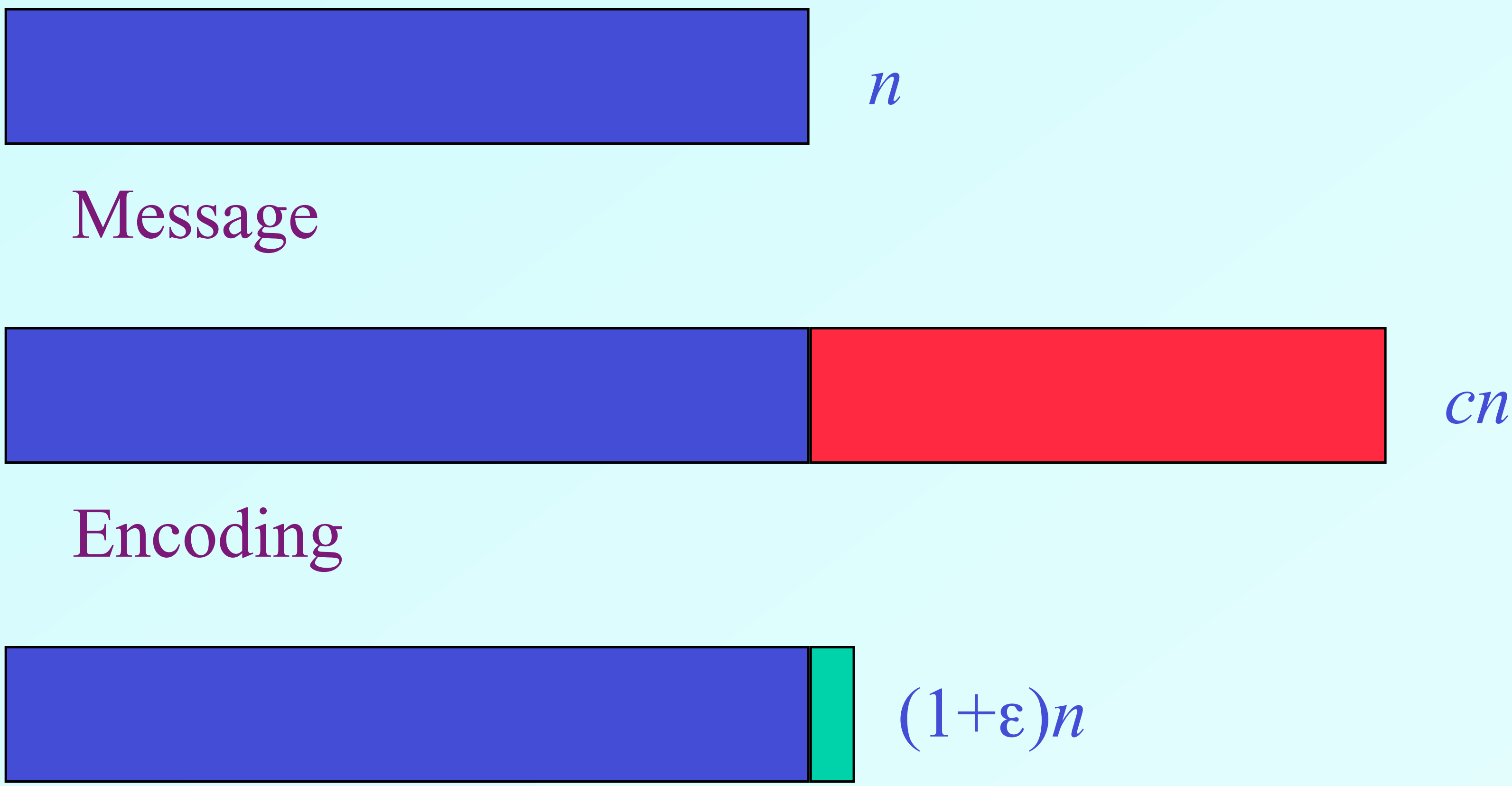
Performance Measures

- **Time Overhead**
  - The time to encode and decode expressed as a multiple of the encoding length.
- **Reception efficiency**
  - Ratio of packets in message to packets needed to decode. Optimal is 1.

Reception Efficiency

- **Optimal**
  - Can decode from any n words of encoding.
  - Reception efficiency is 1.
- **Relaxation**
  - Decode from any $(1+\varepsilon)\ n$ words of encoding
  - Reception efficiency is $1/(1+\varepsilon)$.

Parameters of the Code



Reception efficiency is $1/(1+\varepsilon)$

Previous Work

- **Reception efficiency is 1.**
  - Standard Reed-Solomon
    - Time overhead is number of redundant packets.
    - Uses finite field operations.
  - Fast Fourier-based
    - Time overhead is $\ln^2 n$ field operations.
- **Reception efficiency is $1/(1+\varepsilon)$.**
  - Random mixed-length linear equations
    - Time overhead is $\ln(1/\varepsilon)/\varepsilon$.

Tornado Code Performance

- **Reception efficiency is $1/(1+\varepsilon)$.**
- **Time overhead is $\ln(1/\varepsilon)$.**
- **Simple, fast, and practical.**

Codes: Other Applications?

- **Using codes, data is like water.**
- **What more can you do with this idea?**
- **Example --Parallel downloads: Get data from multiple sources, *without the need for co-ordination*.**

Bloom Filters: High Level Idea

- **Everyone thinks they need to know exactly what everyone else has.** ***Give me a list of what you have.***
- **Lists are long and unwieldy.**
- **Using Bloom filters, you can get small, approximate lists.** ***Give me information so I can figure out what you have.***

Lookup Problem

- **Given a set $S = \{x_1, x_2, x_3, \ldots x_n\}$ on a universe U, want to answer queries of the form:**

$$Is\ y \in S.$$

- **Example: a set of URLs from the universe of all possible URL strings.**
- **Bloom filter provides an answer in**
  - "Constant" time (time to hash).
  - Small amount of space.
  - But with some probability of being wrong.

Bloom Filters

Start with an m bit array, filled with 0s.

B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hash each item x_j in S k times. If $H_i(x_j) = a$, set $B[a] = 1$.

B	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

To check if y is in S, check B at $H_i(y)$. All k values must be 1.

B	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

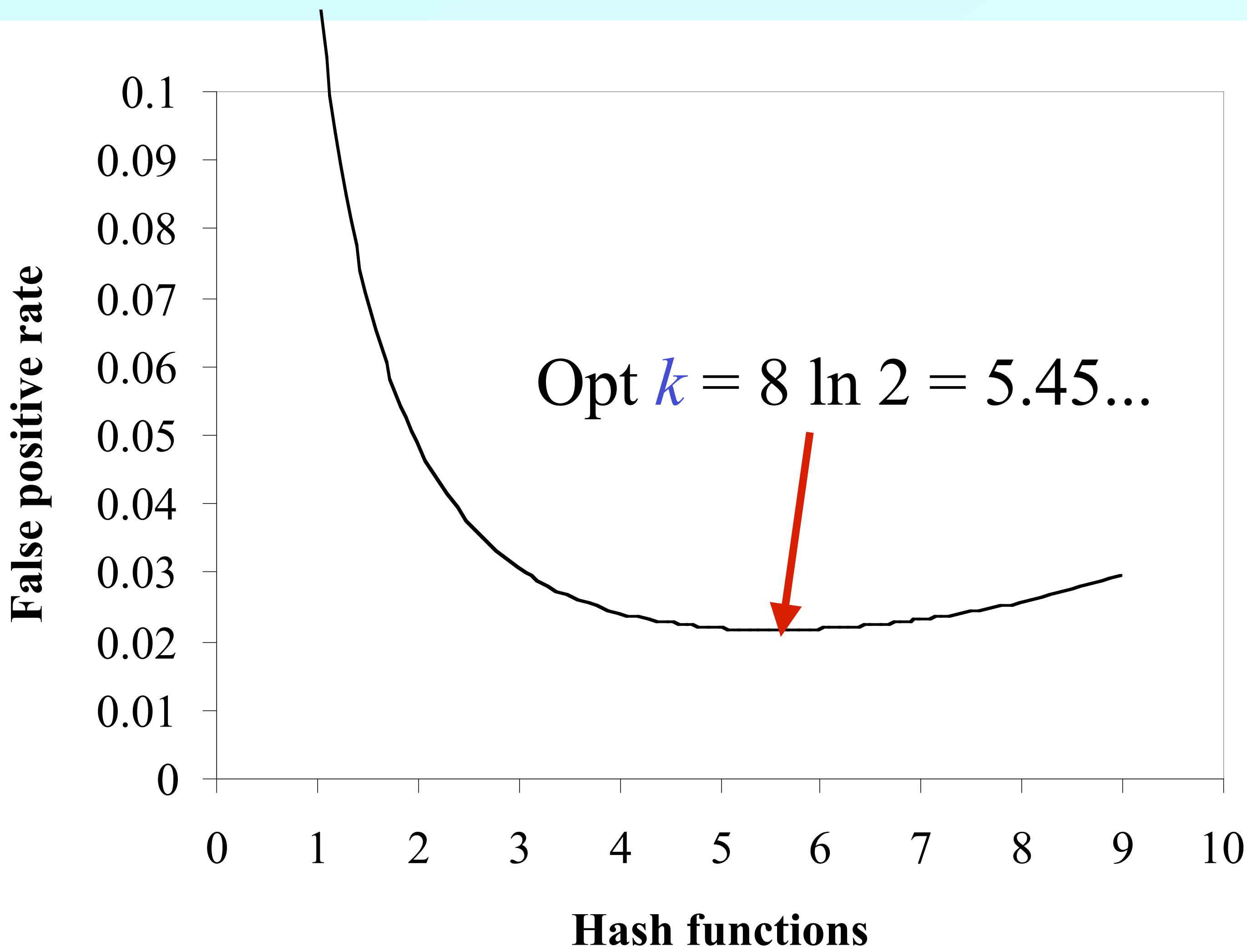
Possible to have a false positive; all k values are 1, but y is not in S.

B	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

Errors

- **Assumption: We have good hash functions, look random.**
- **Given *m* bits for filter and *n* elements, choose number *k* of hash functions to minimize false positives:**
  - Let $p = \Pr[\text{cell is empty}] = (1 - 1/m)^{kn} \approx e^{-kn/m}$
  - Let $f = \Pr[\text{false pos}] = (1 - p)^{k} \approx (1 - e^{-kn/m})^{k}$
- **As *k* increases, more chances to find a 0, but more 1's in the array.**
- **Find optimal at $k = (\ln 2)m/n$ by calculus.**

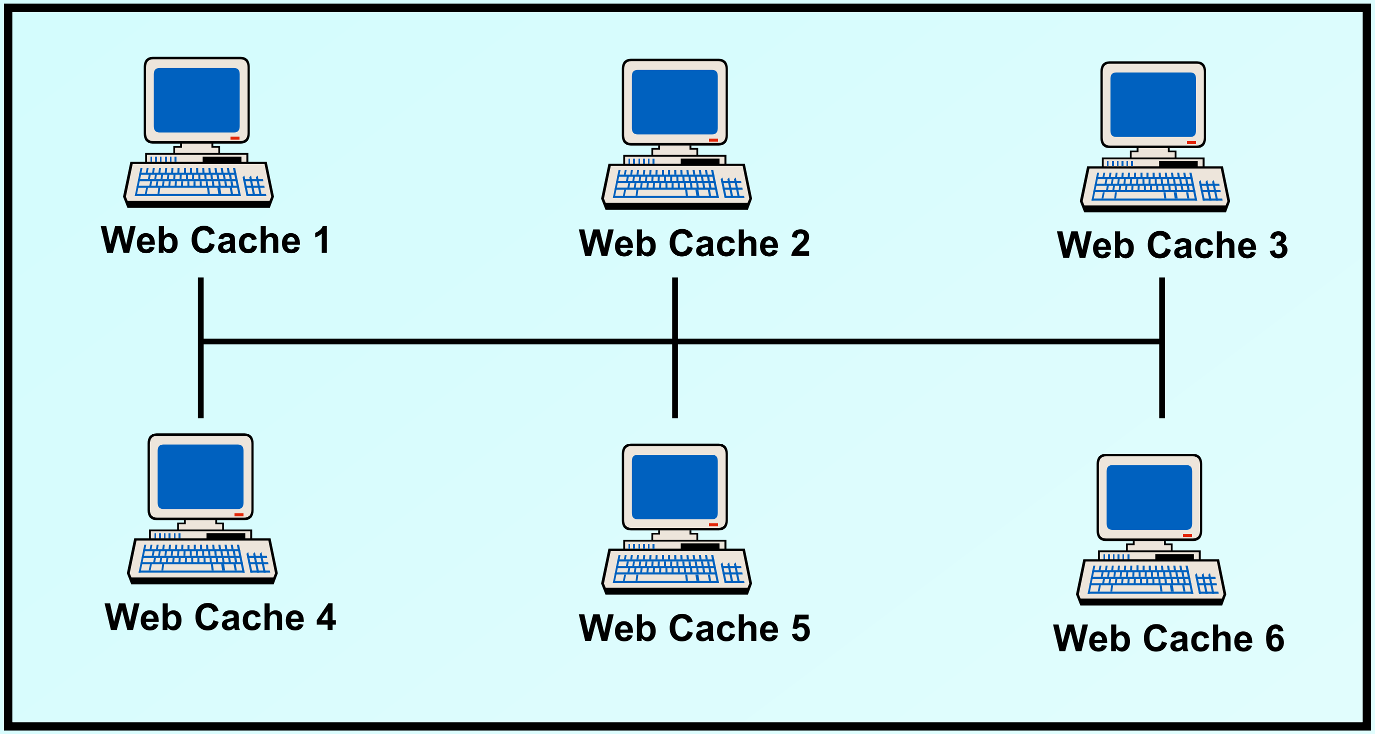
Example



False positive rate
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
1
2
3
4
5
6
7
8
9
10
Hash functions
Opt k = 8 ln 2 = 5.45...

$m/n = 8$

Bloom Filters: Distributed Systems



- **Send Bloom filters of URLs.**
- **False positives do not hurt much.**
  - Get errors from cache changes anyway.

Tradeoffs

- **Three parameters.**
  - Size m/n : bits per item.
  - Time k : number of hash functions.
  - Error f : false positive probability.

Compression

- **Insight: Bloom filter is not just a data structure, it is also a message.**
- **If the Bloom filter is a message, worthwhile to compress it.**
- **Compressing bit vectors is easy.**
  - Arithmetic coding gets close to entropy.
- **Can Bloom filters be compressed?**

Optimization, then Compression

- **Optimize to minimize false positive.**

$$p = \Pr[\text{cell is empty}] = (1 - 1/m)^{kn} \approx e^{-kn/m}$$

$$f = \Pr[\text{false pos}] = (1 - p)^{k} \approx (1 - e^{-kn/m})^{k}$$

$$k = (m \ln 2)/n \text{ is optimal}$$

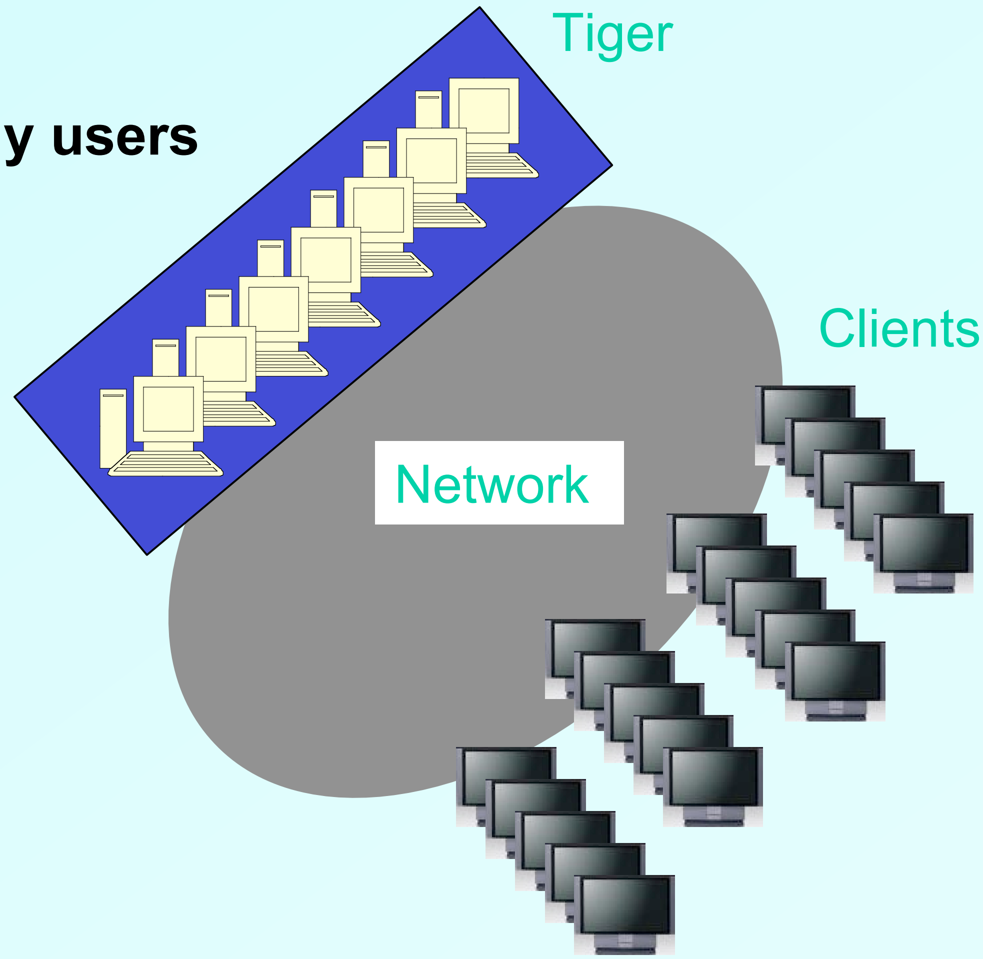
- **At $k = m (\ln 2)/n$, $p = 1/2$.**
- **Bloom filter looks like a random string.**
  - Can't compress it.

Bloom Filters: Other Applications?

- **Finding objects**
  - Oceanstore : Object Location
  - Geographical Region Summary Service
- **Data summaries**
  - IP Traceback

Tiger design goals

- **Video on demand for many users**
- **Quality of service**
- **Scalable and distributed**
- **Low cost hardware**
- **Fault tolerant**

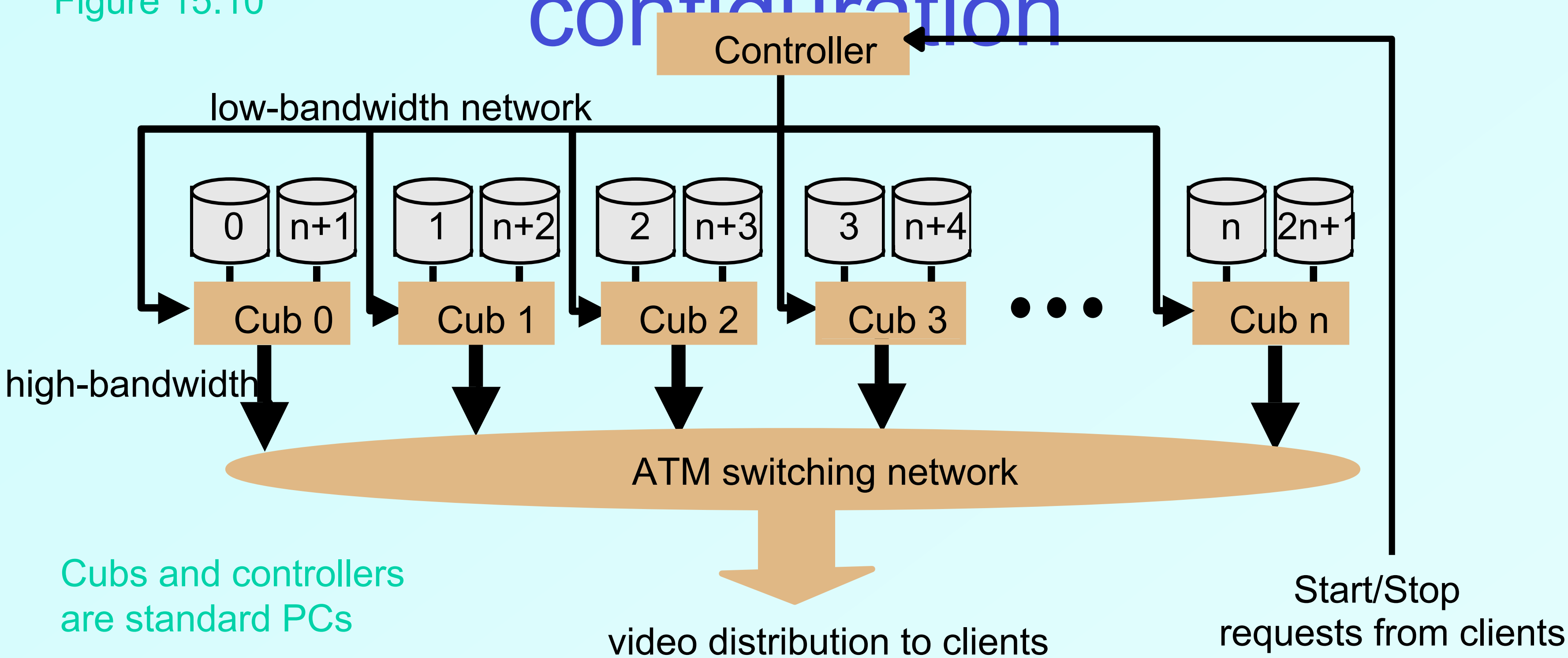


Tiger architecture

- **Storage organization**
  - Striping
  - Mirroring
- **Distributed schedule**
- **Tolerate failure of any single computer or disk**
- **Network support**
- **Other functions**
  - pause, stop, start

Tiger video file server hardware configuration

Figure 15.10

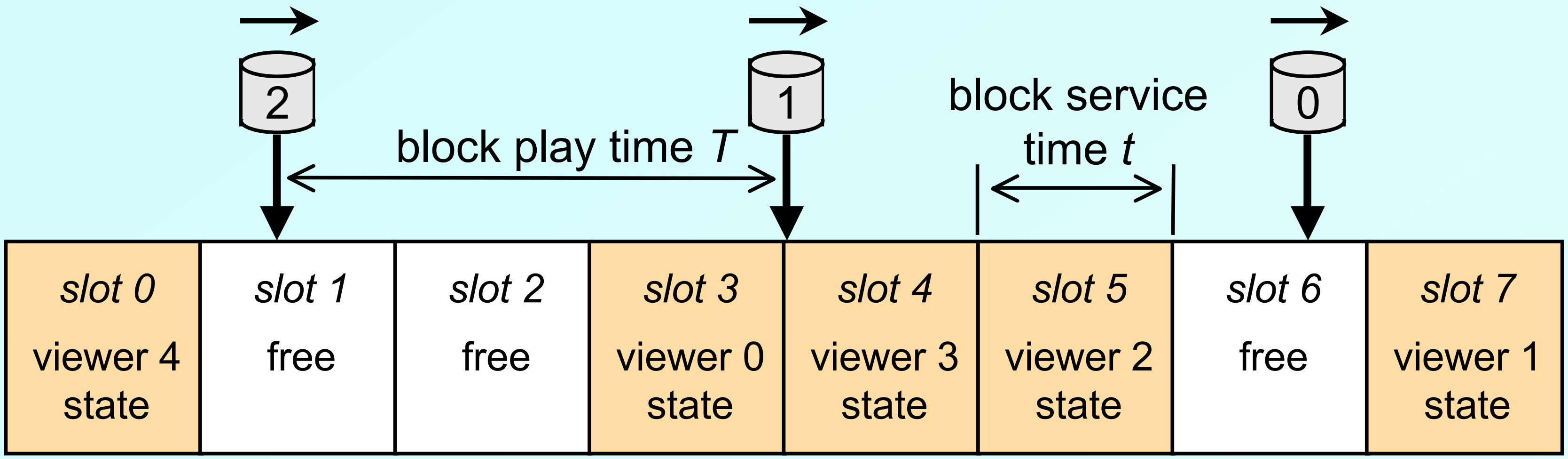


Cubs and controllers are standard PCs

Each movie is stored in 0.5 MB blocks (~7000) across all disks in the order of the disk numbers, wrapping around after n+1 blocks.

Block i is mirrored in smaller blocks on disks i+1 to i+d where *d* is the decluster factor

Tiger schedule



Cub algorithm:

1. Read the next block into buffer storage at the Cub.
2. Packetize the block and deliver it to the Cub's ATM network controller with the address of the client computer.
3. Update viewer state in the schedule to show the new next block and play sequence number and pass the updated slot to the next Cub.
4. Clients buffer blocks and schedule their display on screen.

Tiger performance and scalability

1994 measurements:

– 5 x cubs: 133 MHz Pentium Win NT, 3 x 2Gb disks each, ATM network.

– supported streaming movies to 68 clients simultaneously without lost frames.

– with one cub down, frame loss rate 0.02%

1997 measurements:

– 14 x cubs: 4 disks each, ATM network

– supported streaming 2 Mbps movies to 602 clients simultaneously with loss rate of < .01%

– with one cub failed, loss rate <.04%

The designers suggested that Tiger could be scaled to 1000 cubs supporting 30,000 clients.

Discussion

- **Do we need Tiger today for building a video file-server?**