algo lma
laboratoire d'algorithmique
laboratoire de mathematiques algorithmique

Raptor Codes

Amin Shokrollahi

EPFL
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Swisscom, January 2006

Synopsis

1. Some data Transmission Problems and their (conventional) solutions

2. Fountain Codes
   - 2.1. Definition
   - 2.2. Some type of fountain codes
   - 2.3. LT-Codes
   - 2.4. Raptor Codes
   - 2.5. Systematic codes

3. Standards activities

4. Demo

Point-to-Point Communication

Software Company XYZ with main office in Sillicon Valley wants to exchange gigabytes of data daily with its branch office in India.

XYZ leases a guaranteed 10mbps link, and thinks that this way K gigabytes of data are transmitted in

$$\frac{K \times 2^{33}}{10^7}$$

seconds using standard software, like ftp.

In reality, however, the transfer time is about 10 times slower. Productivity falls.

Facit: XYZ may be a great software company, but it has poor knowledge of networking!

What happened?

Point-to-Point Communication

ftp, like many other applications, uses the ubiquitous transmission control protocol (TCP).

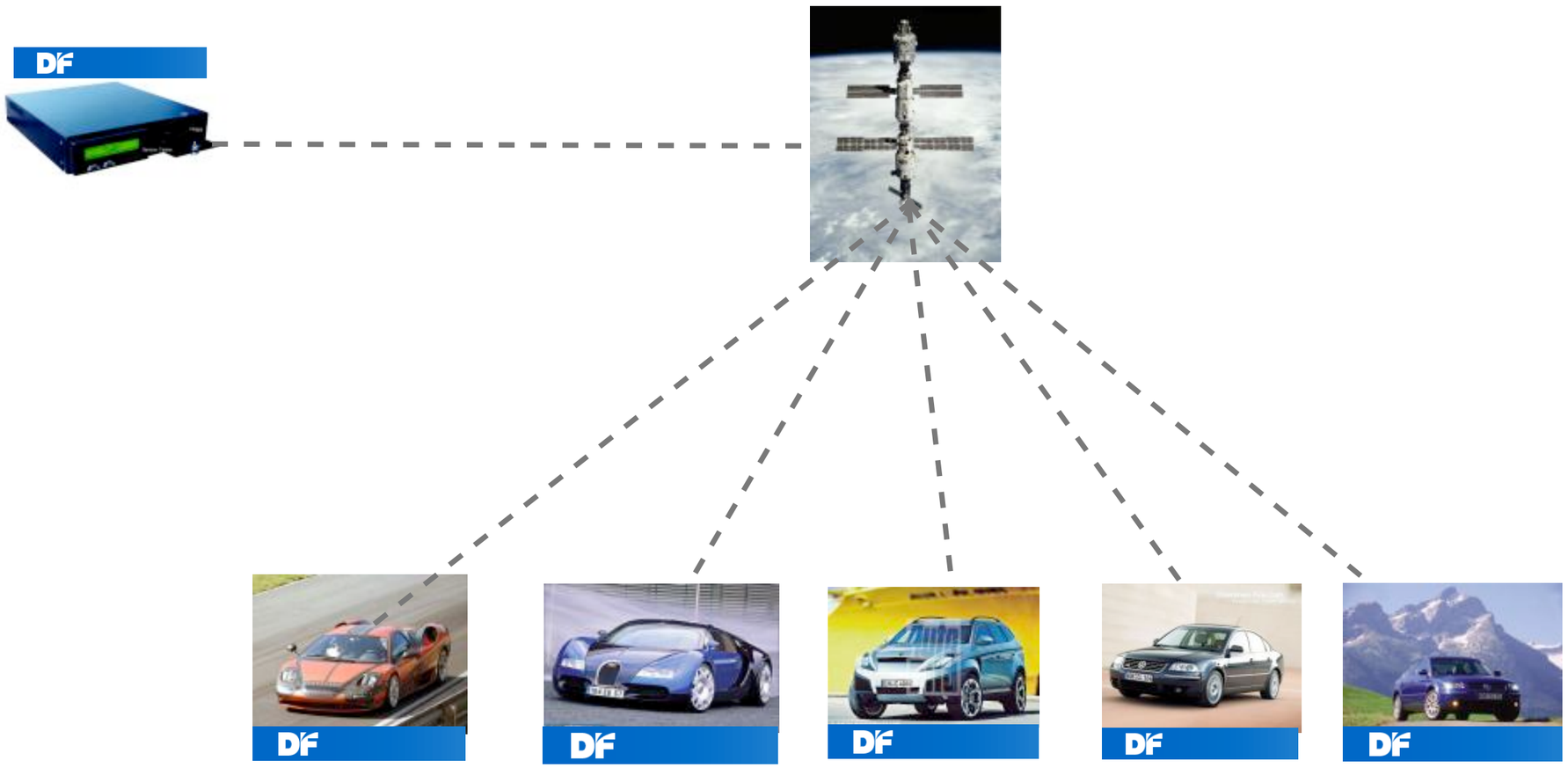
In essence, TCP needs acknowledgement of each packet to guarantee reliable delivery. A lot of time is spent on waiting for acknowledgements.

The actual throughput of TCP (in its steady state) is inversely proportional to the round-trip-time (RTT) between the sender and the receiver.

The RTT puts an upper bound on the transmission speed of TCP, no matter the size of the leased line is! The RTT between California and Cupertino dictates a transmission speed of about 1mbps, even if a lot more is available.

XYZ needs to find a different solution to increase productivity!

Point-to-Multipoint Communication



Challenge: Cars see satellite at completely random times and experience massive amounts of loss. Moreover, no feedback exists between satellite and cars.

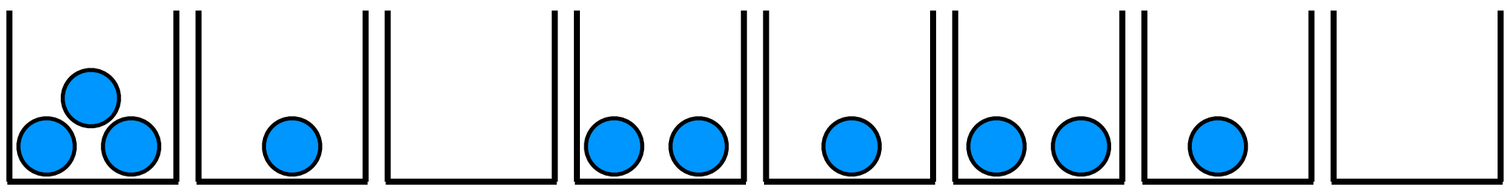
Point-to-Multipoint Communication

Trivial solution: send the original data several times in a carousel manner.

Original file consists of k packets; cars tune in at random times, and each time they receive b packets.

Assume that a complete transmission takes one day, that every car tunes in 2 times per day. How many rounds t of transmission are needed to ensure that 99.99% of the cars have received all the packets? (Minimum is $k/2b$.)

Throw tk balls at random into k bins. For a given bin, what is the probability that it has received at least one ball?



k packets

Point-to-Multipoint Communication

In every round, every bin receives a ball with probability $2b/k$. (This is a good approximation when b is much smaller than k.)

Probability that it is empty after t rounds is

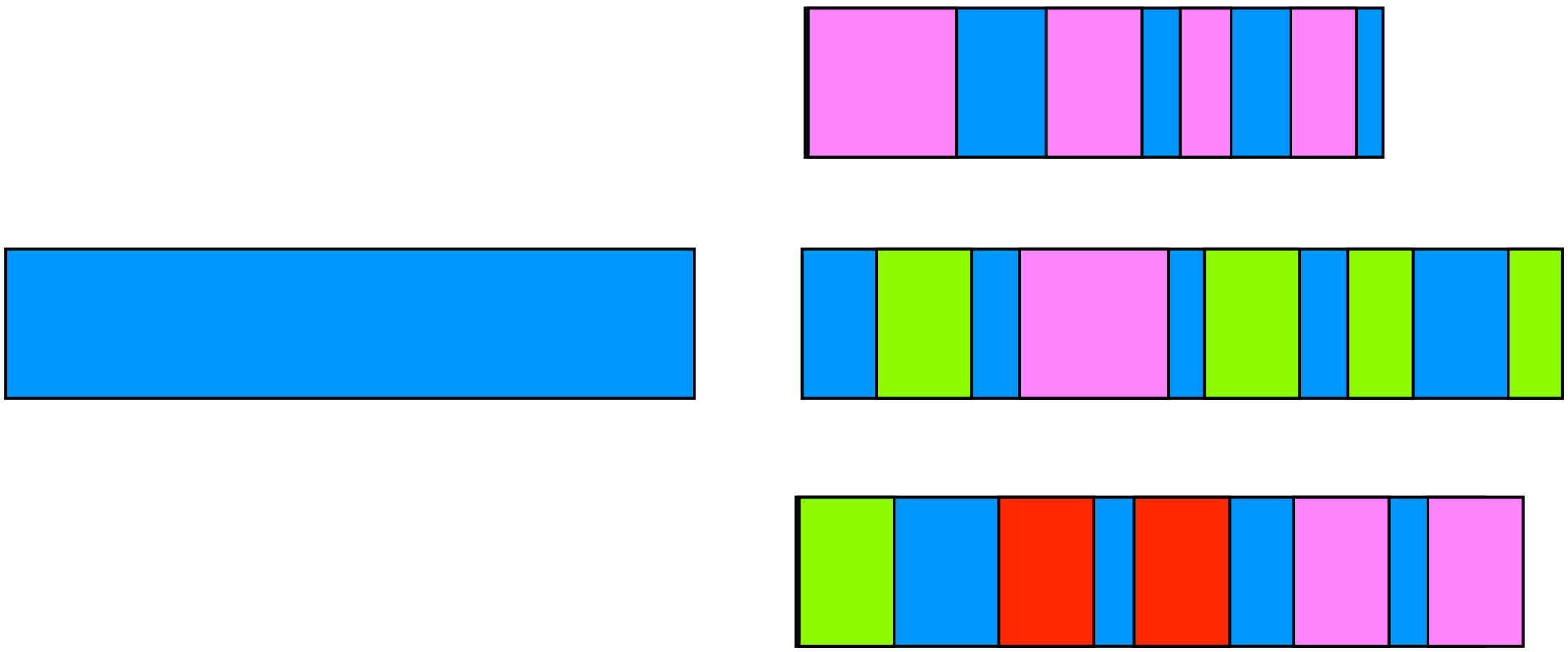
$$\left(1 - \frac{2b}{k}\right)^t \sim \mathrm{e}^{-2bt/k}$$

Want this quantity to be less than 0.0001; so t is roughly $4.6k/2b$. This means that every car has received 9.4 k packets (instead of only k) of which many are duplicate. Precious bandwidth has been wasted.

In reality (for a more realistic model) the value of t is larger. Moreover, what we want is a method that ensures that each car can recover the original data after receiving roughly only k packets.

Point-to-Multipoint Communication

Software update for many clients:



Want download time for each client to be proportional to the difference between their versions and the newest version and that master is capable of supporting an potentially unlimited number of clients.

We are not aware of good standard solutions to this problem.

Point-to-Multipoint Communication

Rise of 3G wireless will lead to multicast/broadcast services for multimedia content.

Conventional methods like TCP will lead to massive overheads in bandwidth consumption.

The user-datagram-protocol (UDP) leads to unreliable delivery and is unusable (so customers will not pay for the services).

So does multicast.

Problem must be solved, however, to ensure economical viability of a variety of 3G services.

Multipoint-to-Point Communication

A big software company wants to diversify its retail sites, both for reliable disaster recovery, and for faster service.

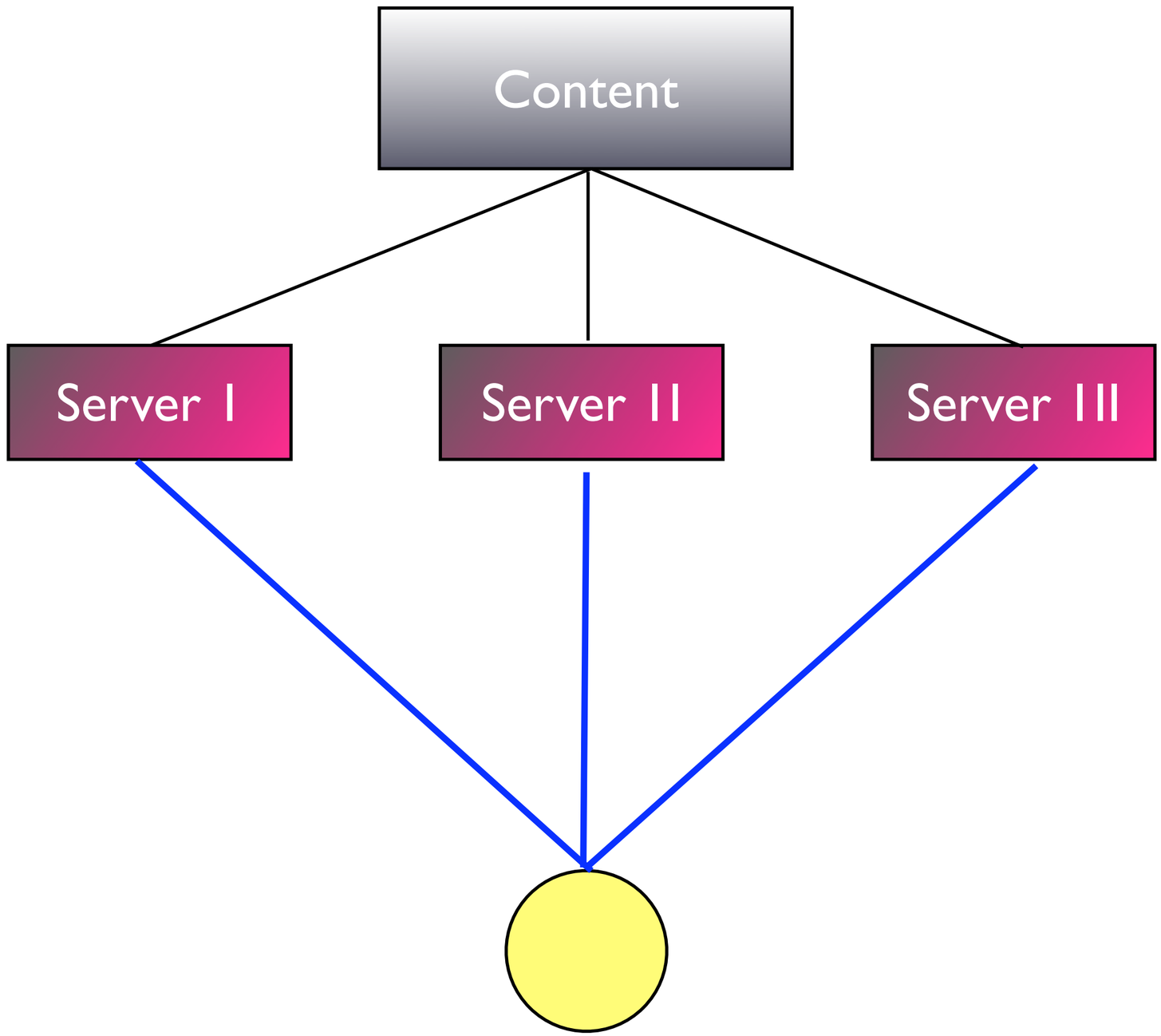
Clients should be able to connect to multiple sites and receive the same data from all sites at once.

The amount of data each client receives should be essentially equal to the size of the requested file.

Duplicate copies of the same portion of the file should thus be avoided.

How can this be done with minimial, or better, no central management?

Multipoint-to-Point Communication



Multipoint-to-Point Communication

Downloading popular files in a P2P network:

Want to be able to connect to multiple users, and download the file at once from all of them.

Download should have minimal (or almost no) management overhead.

It should be robust to users transiently joining and leaving the P2P network.

Many current systems do this by segmenting the file and requesting each segment from a different user, possibly at entirely different rates.

Tradeoff: need small segments so to be able to download from many users at once. On the other hand, need large segments to avoid negotiation and management overhead.

Multipoint-to-Multipoint Communication

Path diversity:

Want to send data to multiple sites over unreliable links.

Examples:

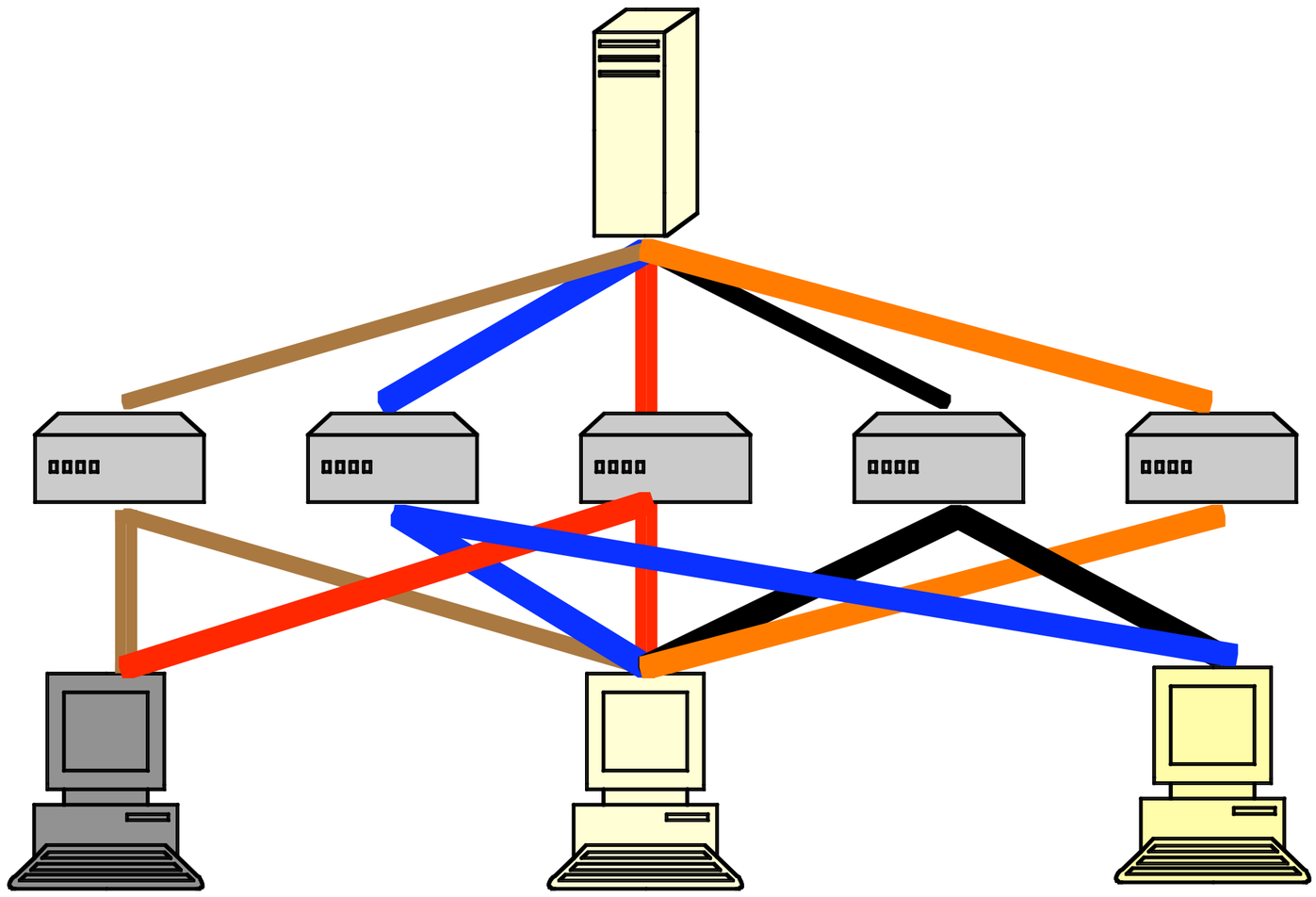
Download of data in rural areas, through satellite, wireless, and fixed line, all at once.

Download of information in battle field, where each individual line can be intentionally jammed.

Clients should be able to recover the file as soon as they have received an amount of data that is essentially the same as the file.

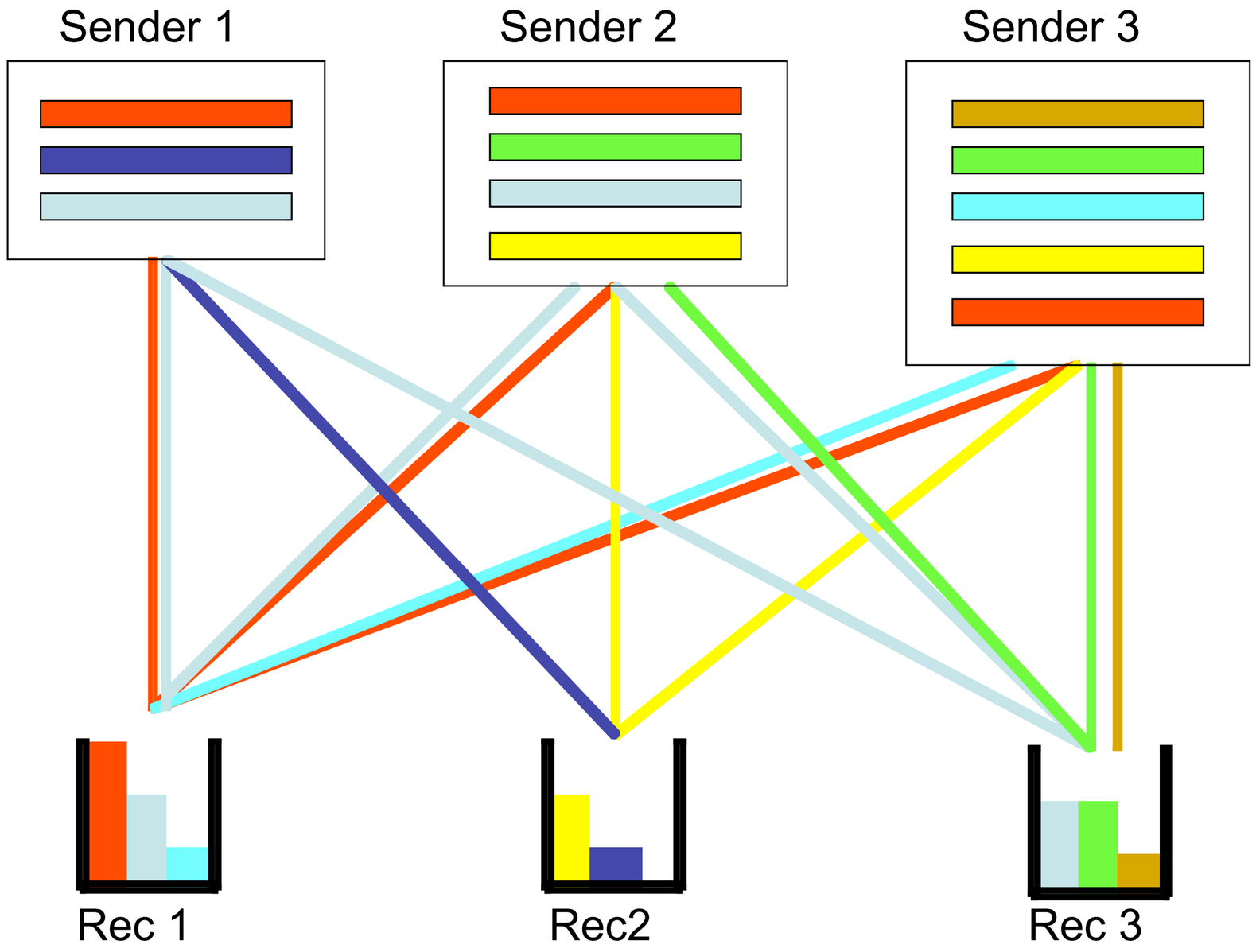
We do not know of any conventional method that can solve this problem efficiently.

Multipoint-to-Multipoint Communication



Multipoint-to-Multipoint Communication

P2P, again:



The Fountain Code Paradigm

How can we solve all these seemingly different problems at once?

Fountain codes provide a framework for such a solution.

Given a piece of data, consisting of k symbols (which can be packets or bits), a fountain code produces a potentially limitless stream of output symbols with the following properties:

1. Each output symbol is generated randomly (according to some distribution) and independently of every other symbol.

2. It is possible to recover the original k symbols from any set of m received symbols, with high probability, for some m (which is at least k). High probability means a probability of at least $1 - \frac{1}{k^c}$

The quantity m/k-1 is called the overhead of the fountain code.

Why the Name Fountain Code?

Think of the symbols as drops of water.

Fill a digital bucket with these drops.

As soon as you have enough drops, the bucket is full, and you can drink your water.

It does not matter which particular drops fill your bucket; only the total amount matters.

The encoding mechanism is thus like a fountain producing symbols.

Let us now go back and see how fountain codes can solve all the data transmission problems we mentioned.

Point-to-Point Communication

The TCP problem of the software company XYZ:

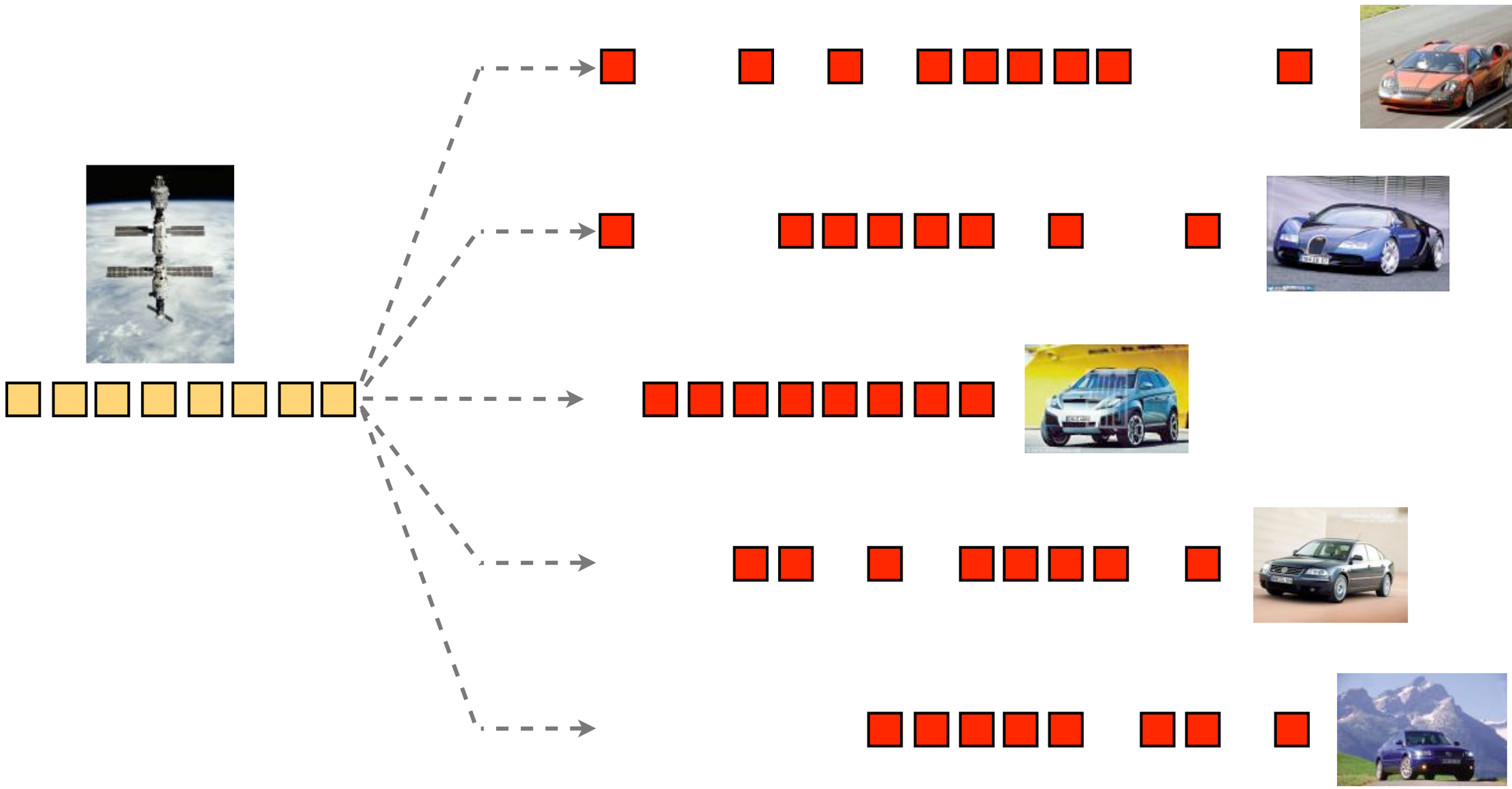
Use a fountain code to produce output packets, and send the packets via UDP rather than TCP.

This way an arbitrary portion of the 10mbps can be filled (provided the encoding process is fast enough).

XYZ can increase productivity again!

The method is adopted by several Hollywood infrastructure companies specialized in delivery of “Digital Dailies.” In fact, this method was used to keep the production of several blockbuster movies such as “The Last Samurai” and “Lord of the Rings: The Return of the King” on schedule.

Point-to-Multipoint Communication



Receivers adjust to their individual loss rates

Point-to-Multipoint Communication

Using the carousel method, the bandwidth used was about 4.6 times larger than the optimal.

Using a fountain code, every car received a "fresh" set of $2b$ packets when it listens in.

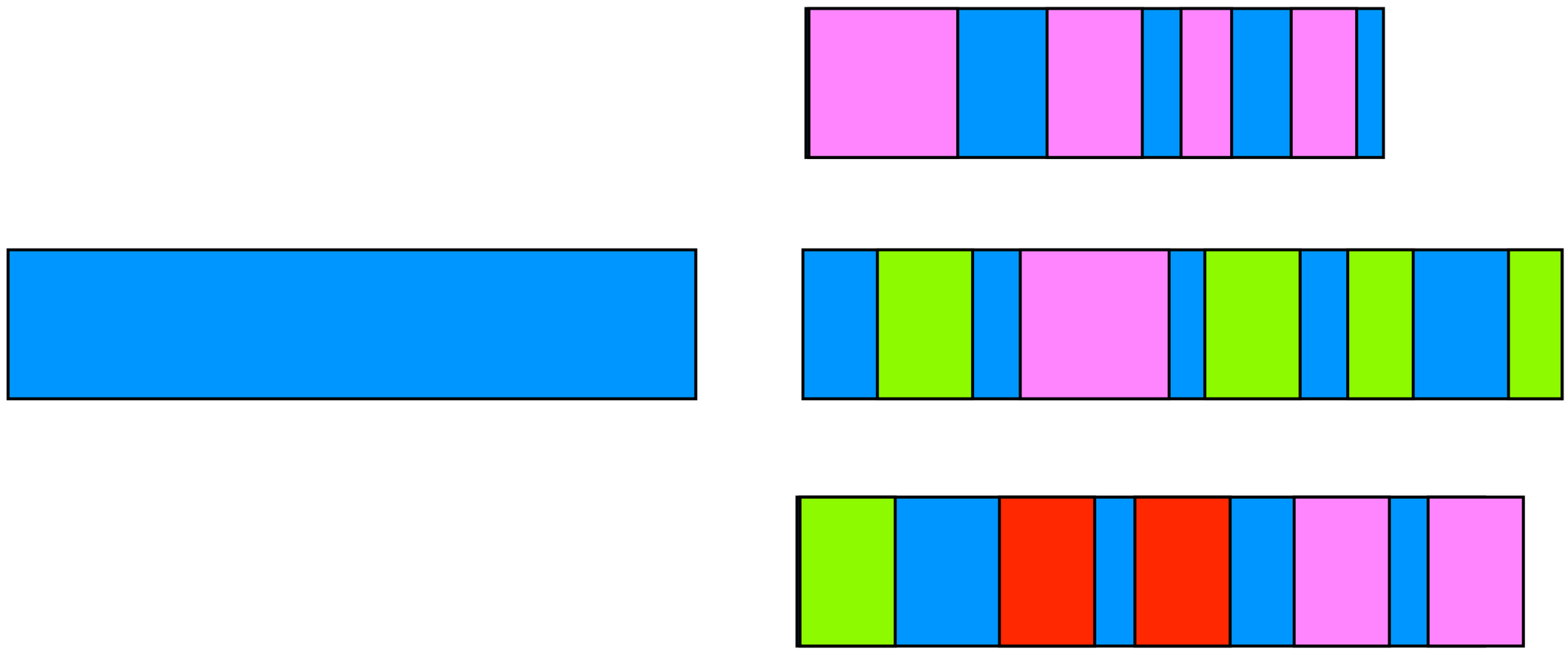
If the fountain code is designed such that recovery of the original file is possible when essentially k packets are received, then after essentially $k/2b$ rounds of transmission every car has enough data to recover the file.

The problem has thus been solved essentially optimally! This is even the case when the model of usage for cars is vastly different from the simple one described earlier.

In fact, two major US satellite radio companies have licensed this method to be able to deliver various types of information to cars via their satellite links. (Same is true for a major Japanese automobile company.)

Point-to-Multipoint Communication

Software update for many clients:



Here a method called “multicast rsync” can be used in combination with “systematic fountain codes” to ensure that every client can recover the latest version of the software by receiving. (Ideas due to James Clarke, and Jack Wolf.)

What is a systematic fountain code?

Systematic Fountain Codes

In a systematic fountain code the original symbols are sent alongside the additional symbols. In addition to the two main properties of a fountain code, a systematic fountain code satisfies a "uniformity" property:

3. Any set of l original symbols and m-l additional symbols are sufficient to recover the original k symbols with high probability.

A systematic fountain code is designed in a different way than a normal fountain code (more later).

Note that many classes of codes with fast decoding algorithms (such as LDPC or IRA codes) do not satisfy property 3.

Point-to-Multipoint Communication

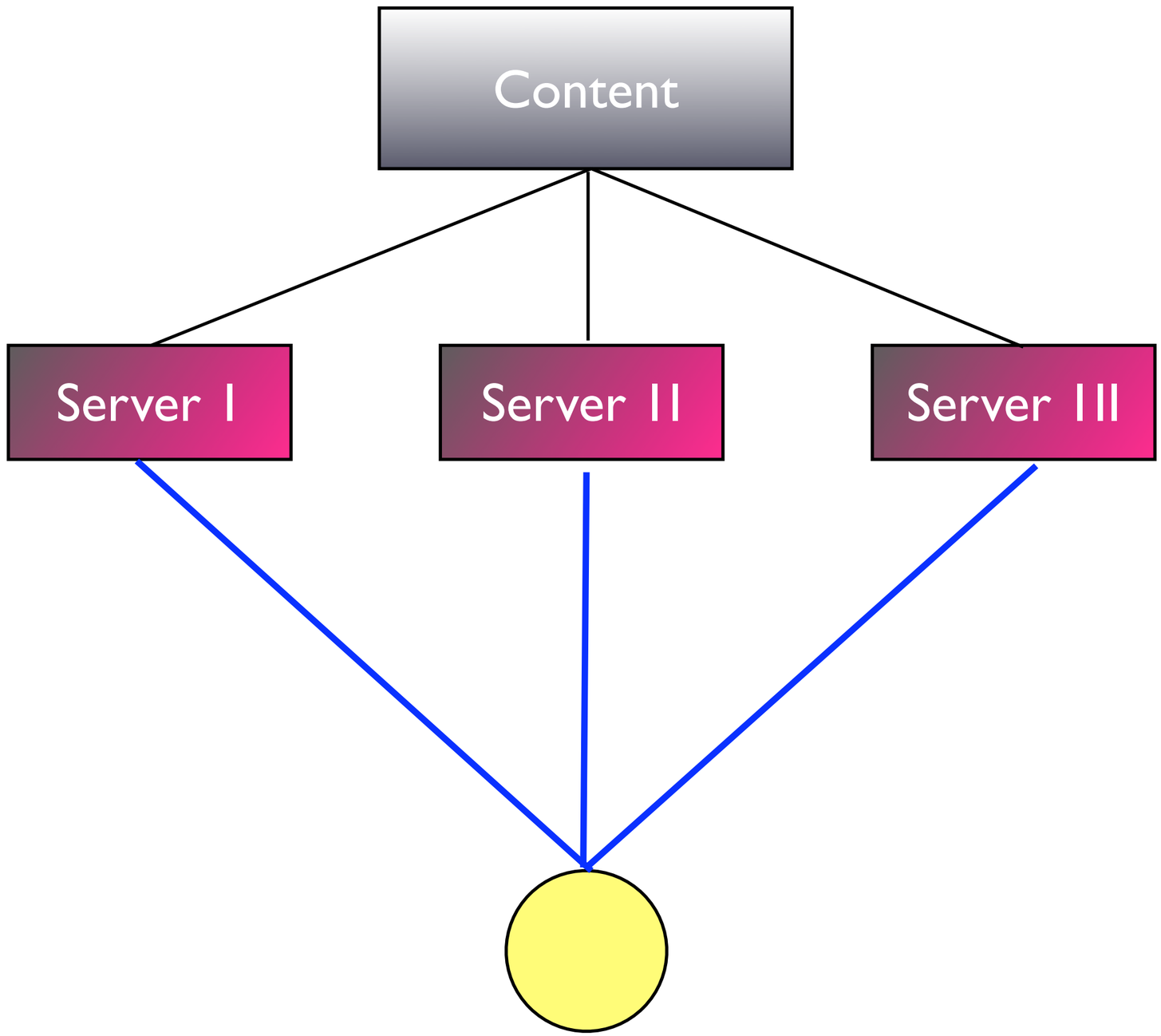
Fountain codes with efficient encoding and decoding algorithms with small overhead can be used in an excellent manner to solve the data distribution problems in upcoming 3G applications, such as rich-media transport.

In fact, a version of fountain codes has been standardized in the 3GPP-MBMS (multicast-broadcast multimedia services) standards body as the sole mandatory standard for data transport.

This means that all future devices supporting these services have to have these codes embedded into them.

We will study these codes and their design later in these lectures.

Multipoint-to-Point Communication



Multipoint-to-Point Communication

Multi-site download applications are a breeze with fountain codes:

Run a fountain encoder on each site. Since the symbols are produced randomly, symbols obtained from the different sites are indistinguishable, and can be considered as coming from only one site.

If a site breaks down or goes off-line for any reason, then this only leads to a slower download, as fewer output symbols are received.

No extra management is necessary to ensure that the client has un-interrupted reception.

Multipoint-to-Point Communication

Downloading popular files in a P2P network:

This is a special case of a multi-site download, and is solvable in much the same way.

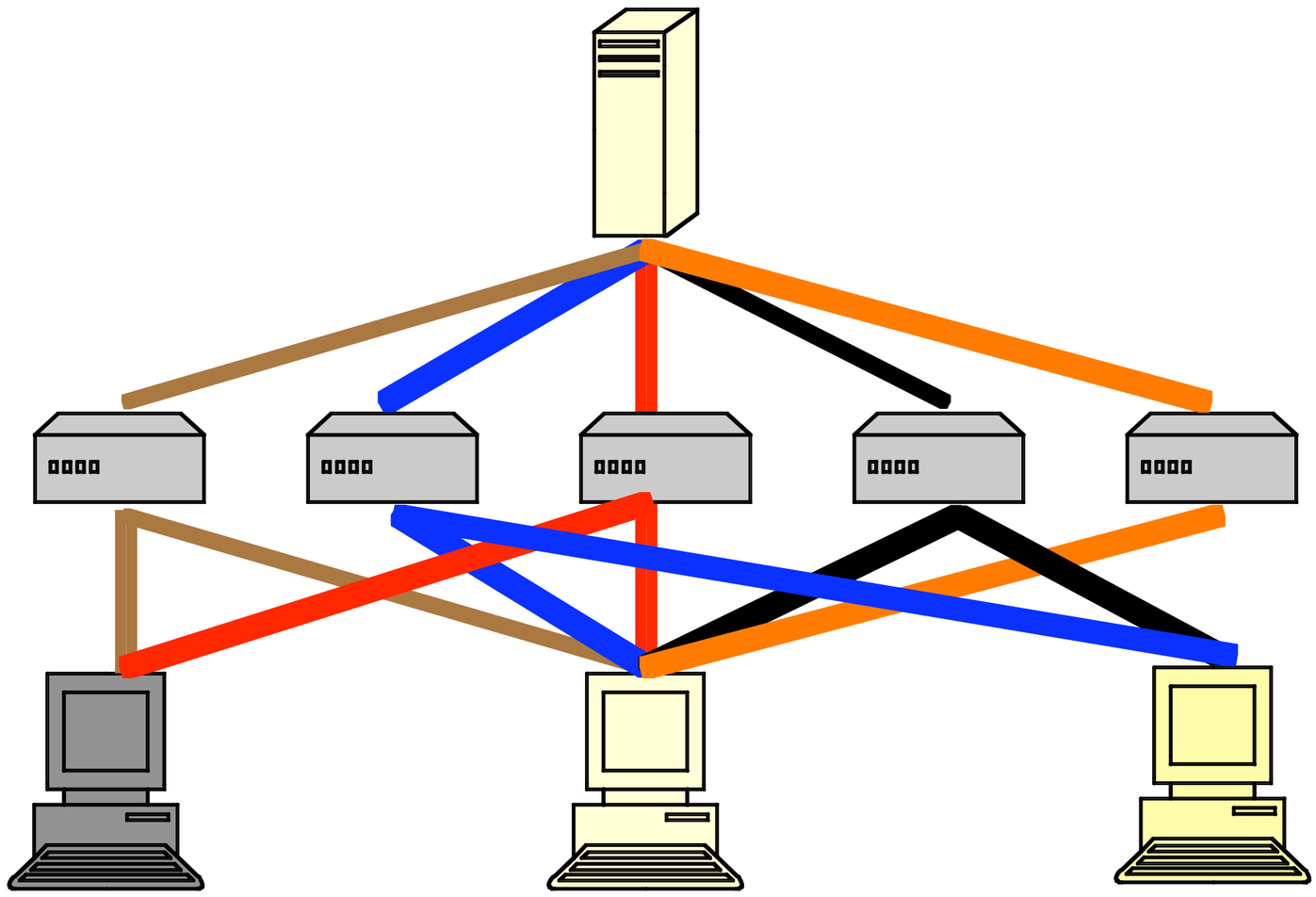
Added benefit: clients in a P2P network can choose the rate at which they want to send the data to the receiver.

The receiver's experience is as if it is receiving from one site at an aggregate rate.

Clients can join or leave without any managment overhead.

Popular movies are particularly interesting, since there are potentially many clients owning a copy, and willing to share it, so the aggregate download speed is usually much larger than non-popular movies.

Multipoint-to-Multipoint Communication



Multipoint-to-Multipoint Communication

This problem can also be solved in the same manner:

The server starts a fountain and sends the output symbols across different paths.

The receiver only needs to collect enough symbols, no matter from which path.

If not all paths are down, the receiver will eventually receive the file.

If the overhead of the fountain code is very small, then the time the receiver needs to receive the file is essentially optimal, no matter how the loss characteristics of the links are!

Summary, and some History

Fountain codes are a class of codes designed for solving various data transmission problems, at the same time.

Fountain codes with fast encoding and decoding algorithms, and (arbitrarily) small overhead are particularly interesting for solving these problems.

Fountain codes were stipulated by Byers et al in 1998, and their applications discussed. A construction was, however, not given.

First construction of efficient Fountain codes was given by Luby (1998, published 2002).

Now, let us start with some theory!

(Binary) Fountain Codes

Fix distribution $\mathcal{D}$ on $(\mathbb{F}_2^k)^*$, where k is the number of input symbols.

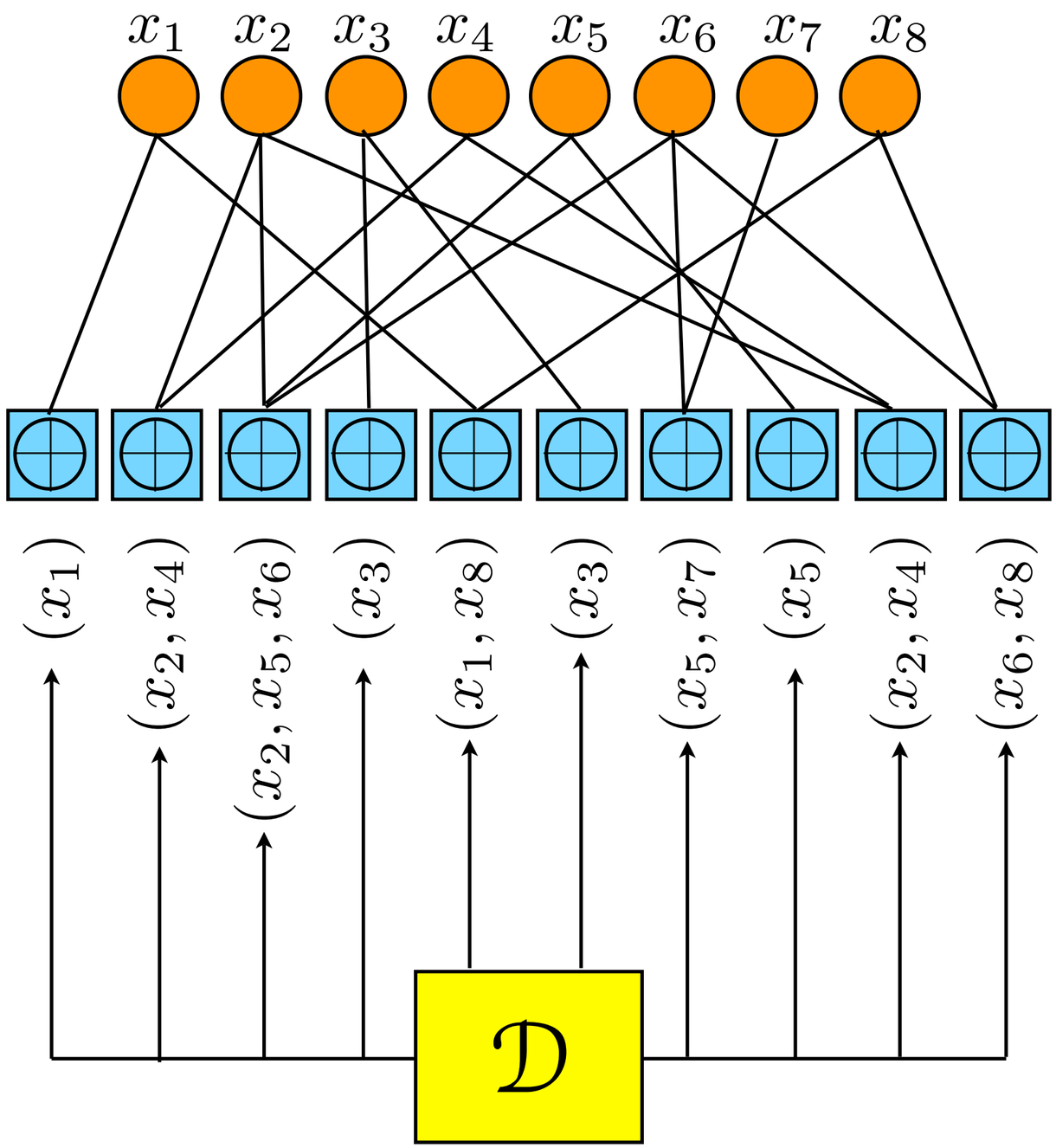
A fountain code with parameters $(\mathcal{D}, k)$ is a vector in $\left((\mathbb{F}_2^k)^*\right)^{\mathbb{N}}$ sampled from $\mathcal{D}^{\mathbb{N}}$.

Distribution $\mathcal{D}$ can be identified with a distribution $\mathcal{D}$ on $\mathbb{F}_2^k$.

Operation:
For each output symbol sample independently from $\mathcal{D}$ and add symbols corresponding to the sampled output.

Symbols are understood to be binary vectors, and additions are understood to be over $\mathbb{F}_2$.

(Binary) Fountain Codes



Important Note

Each sent symbol must contain an indication of how it was generated, otherwise the information in the symbol is useless.

In practice, this can be done by indicating a seed for the random number generator used for the sampling process.

Another possibility is to use a clocking device and shared randomness between sender and receiver.

Encoding Cost

The expected encoding cost of a fountain code with parameters $(\mathcal{D}, k)$ is the expectation of the weight function under $\mathcal{D}$:

$$\mathrm{E}_{\mathcal{D}}[\mathrm{wgt}(x)]$$

This corresponds to the expected per-symbol cost of encoding using the trivial algorithm. Using standard Chernoff bounds it can be shown that the actual average cost is sharply concentrated around this value.

The best one can hope for is that this cost is constant O(1).

Decoding Cost

The expected decoding cost of a fountain code with parameters $(\mathcal{D}, k)$ using a decoding algorithm $\mathcal{A}$ is the expected number of arithmetic operations (additions in $\mathbb{F}_2$) that the algorithm uses to decode the source symbols.

The best one can hope for is that this cost is linear in k, i.e., O(k).

Goals

If decoding can be done from any set of $k(1+\varepsilon)$ output symbols, whp, then we call ε the overhead of the decoder.

Universality: Want sequences of fountain codes for which the overhead is arbitrarily small.

Efficiency: Want sequences of fountain codes and corresponding decoding algorithms with constant encoding and linear decoding cost.

First Example: Random Fountain Codes

Assume that the distribution $\mathcal{D}$ is uniform.

Encoding cost is $k/2$.

Decoding: Collect output symbols, and put them into a matrix; solve system of linear equations.

Example: received symbols (Z_1, Z_2, Z_3, Z_4) correspond to the linear forms

$$(X_1, X_2 + X_3, X_4 + X_1, X_4 + X_3)$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix} = \begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \end{pmatrix}$$

First Example: Random Fountain Codes

What is the overhead? Suppose we collect m output symbols. Success probability is the probability that a random kxm binary matrix has rank k.

This probability is at least $1 - 2^{k-m}$.

If $m = k + c\log(k)$, for some c, then algorithm succeeds with high probability.

Hence, overhead is $\log(k)/k$.

Summary: Random fountain codes have encoding cost $k/2$, decoding cost $O(k^3)$ and overhead $\log(k)/k$ with respect to maximum-likelihood decoding.

This is very far from our goal!

A Large Step Closer: LT Codes

Invented by Michael Luby in 1998.

First class of universal and almost efficient Fountain Codes.

Output distribution has a very simple form.

Encoding and decoding are very simple.

LT Codes

Fix distribution $(\Omega_1, \Omega_2, \ldots, \Omega_k)$ on $\{1, \ldots, k\}$

Distribution $\mathcal{D}$ is given by

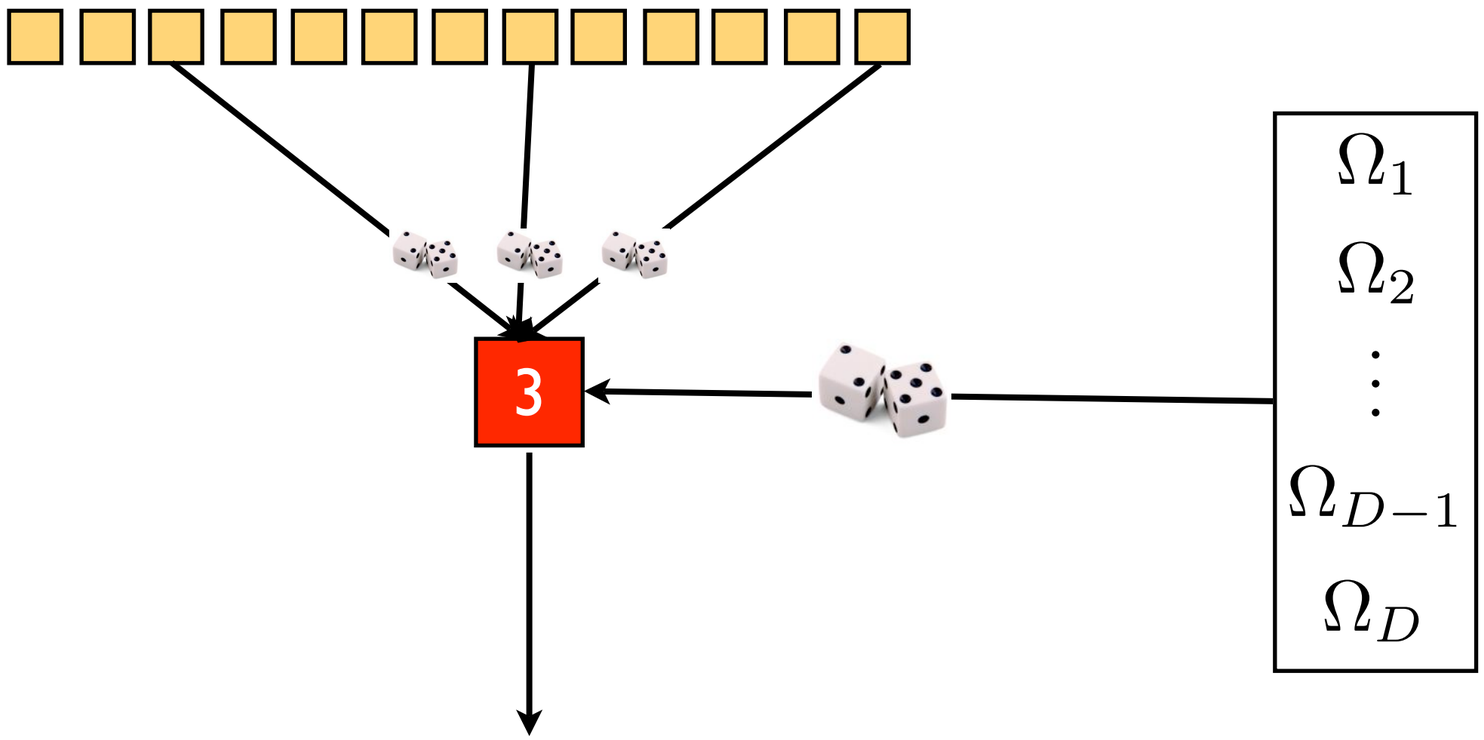
$$\Pr_{\mathcal{D}}(x) = \frac{\Omega_w}{\binom{k}{w}}$$

where w is the Hamming weight of x.

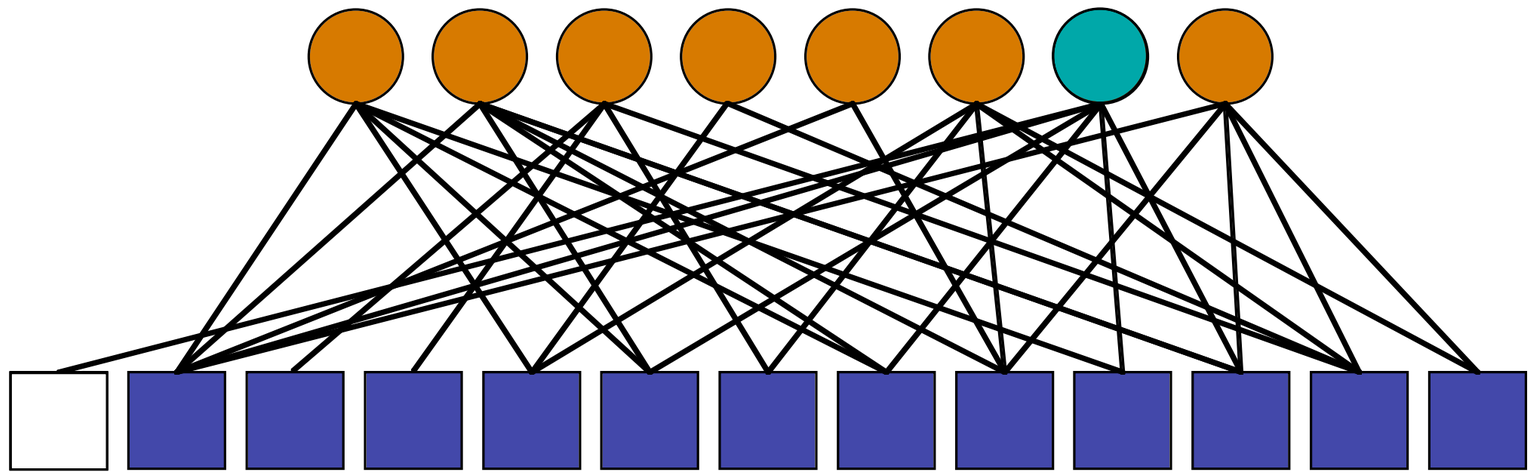
Parameters of the code are $(k, \Omega(x))$

$$\Omega(x) = \sum_{w=1}^{k} \Omega_w x^w$$

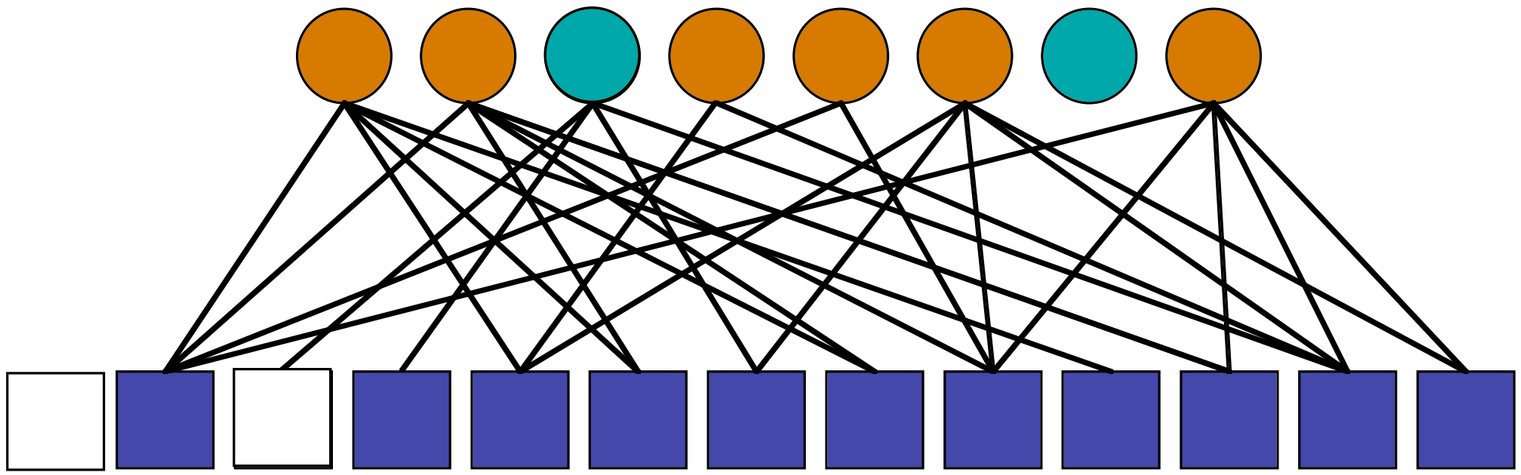
LT Coding Process



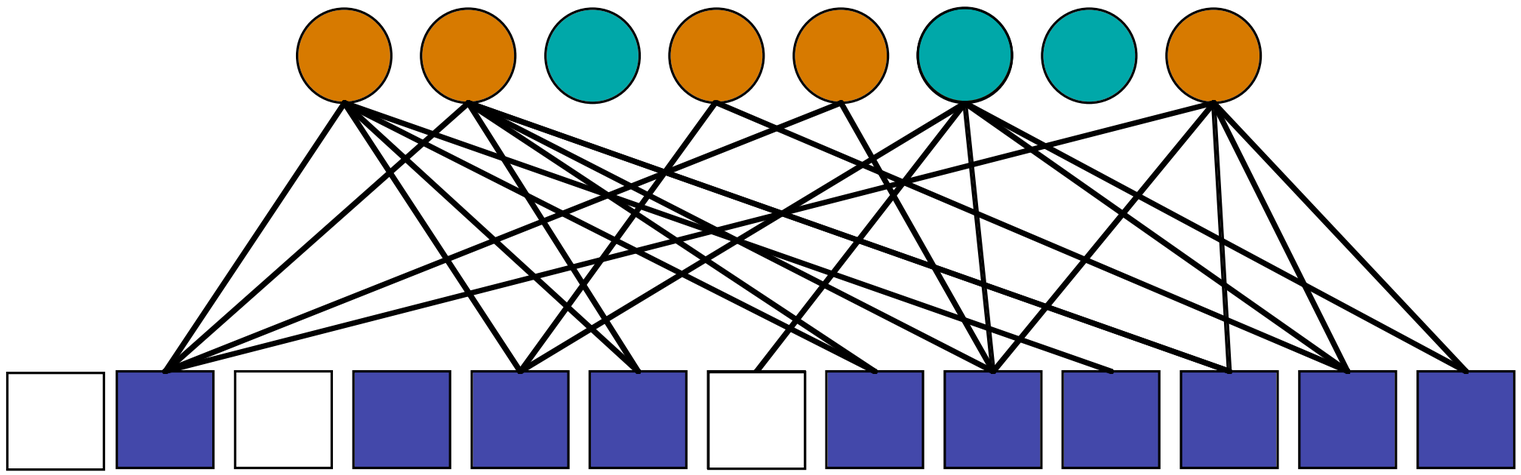
Decoding



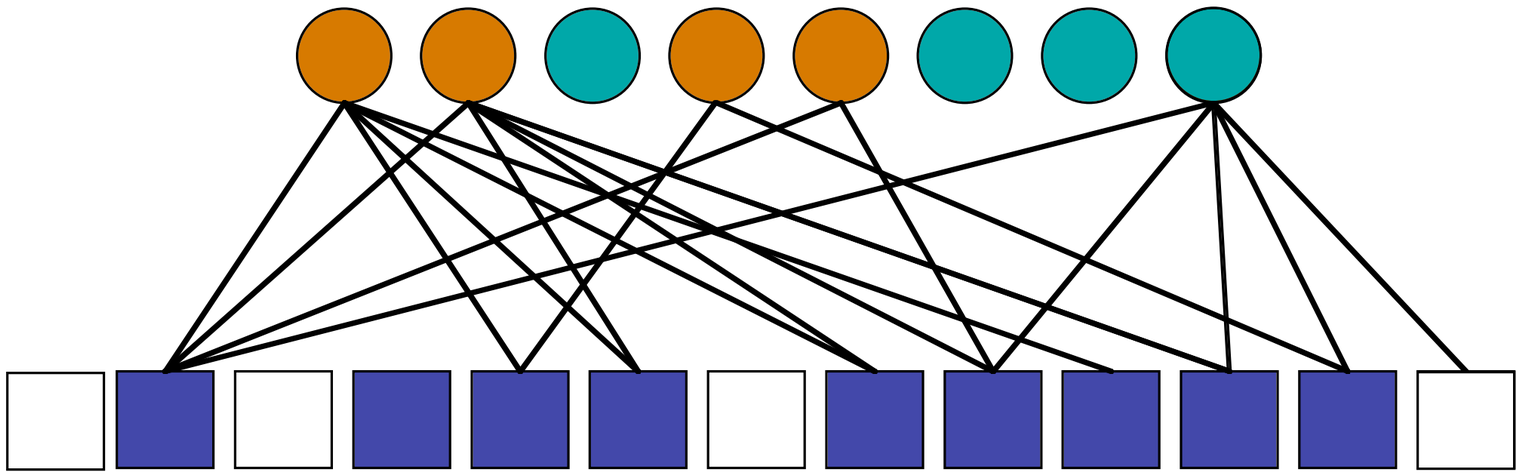
Decoding



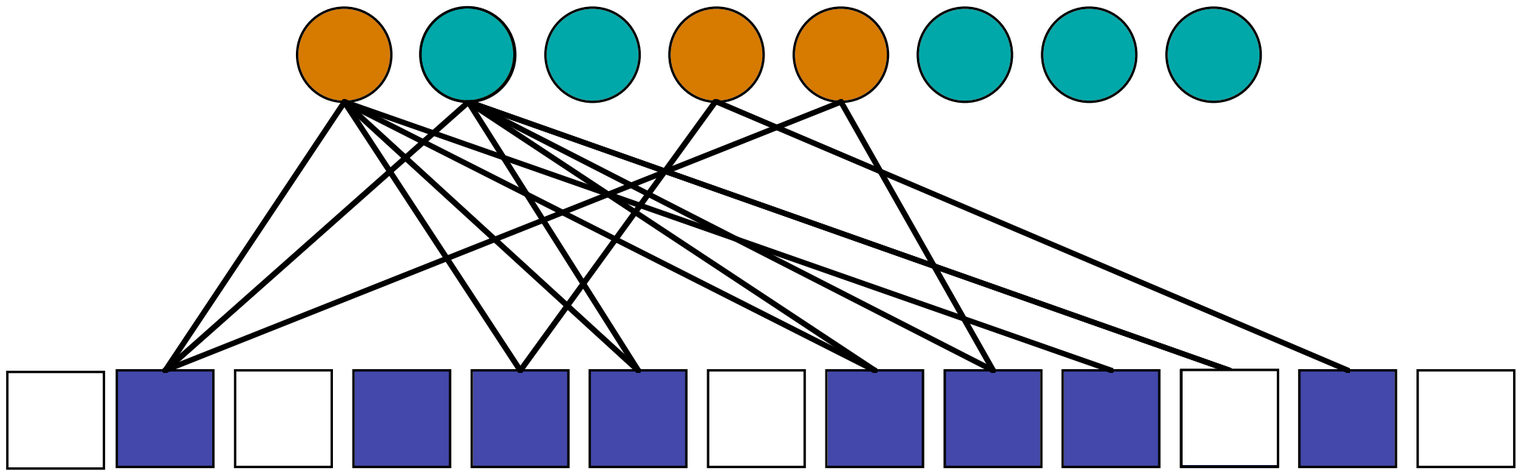
Decoding



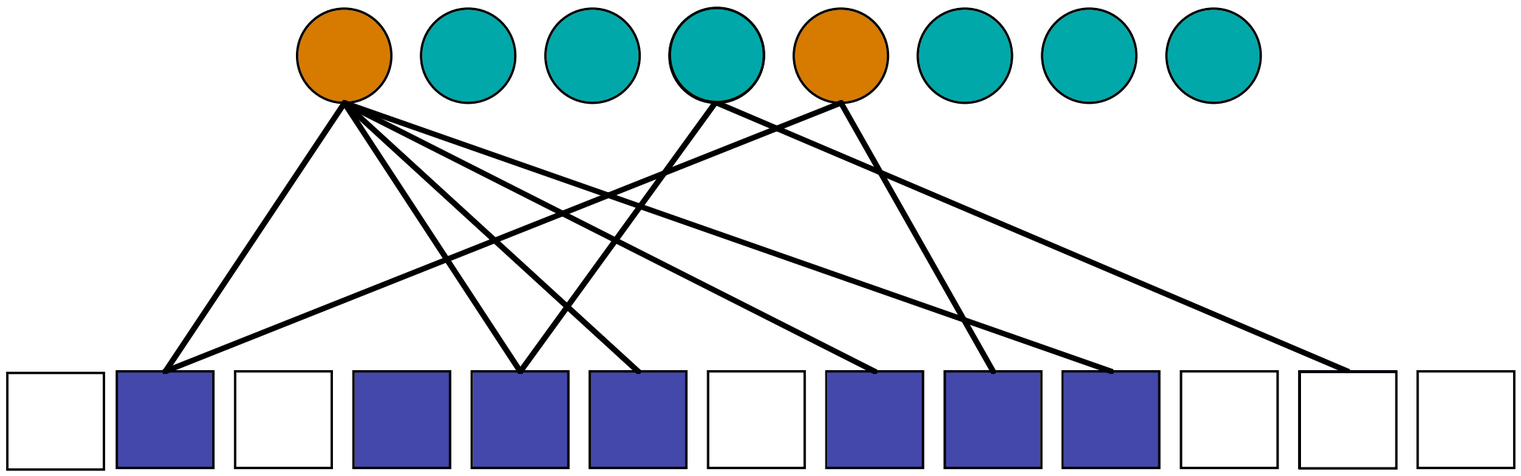
Decoding



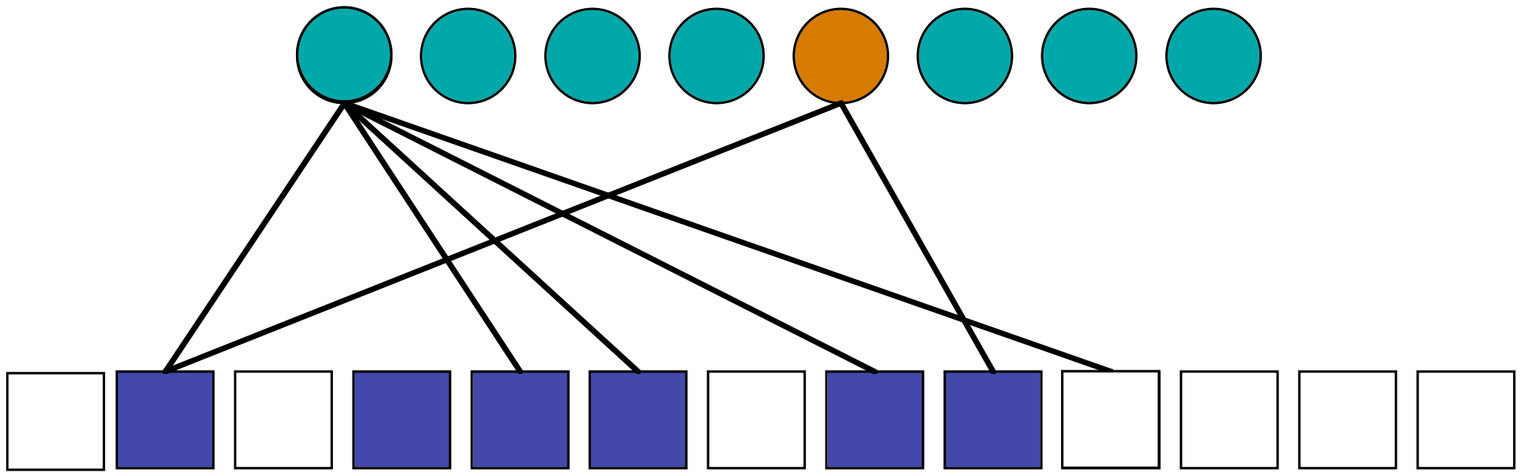
Decoding



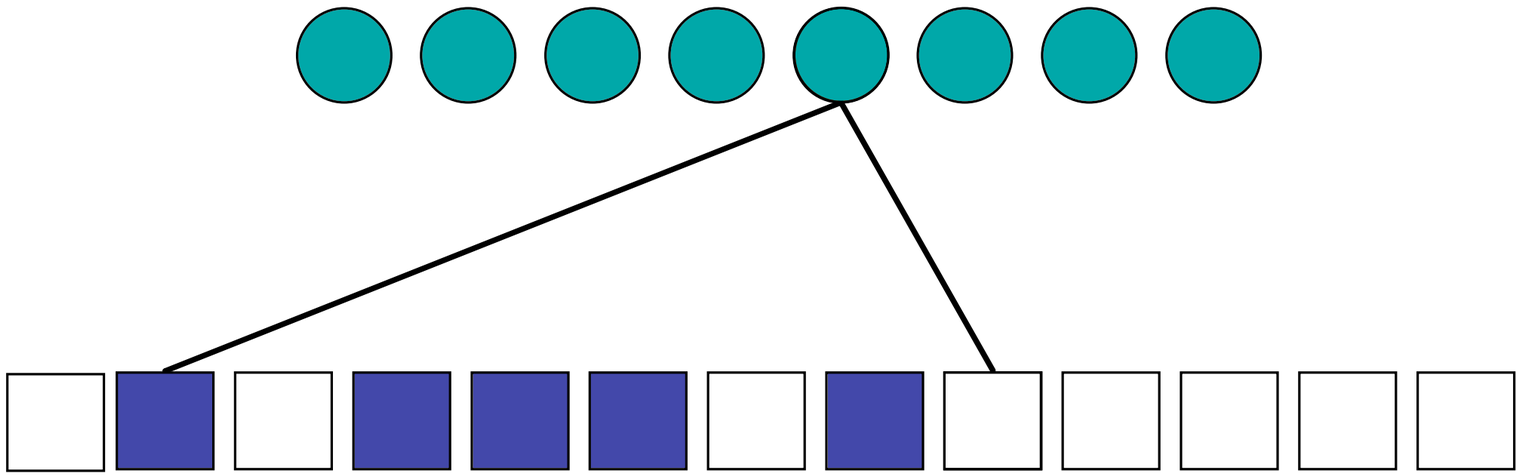
Decoding



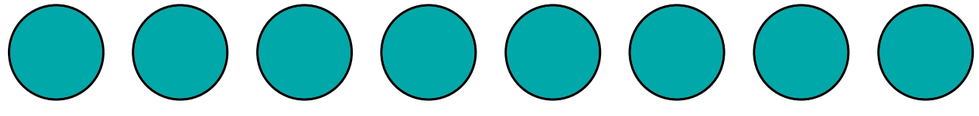
Decoding

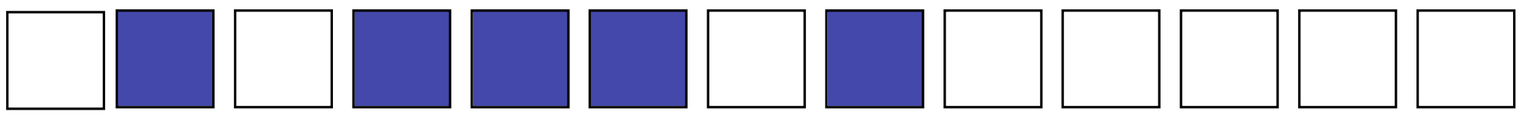


Decoding



Decoding





Parameters

Parameters of an LT-code are $(k, \Omega(x))$.

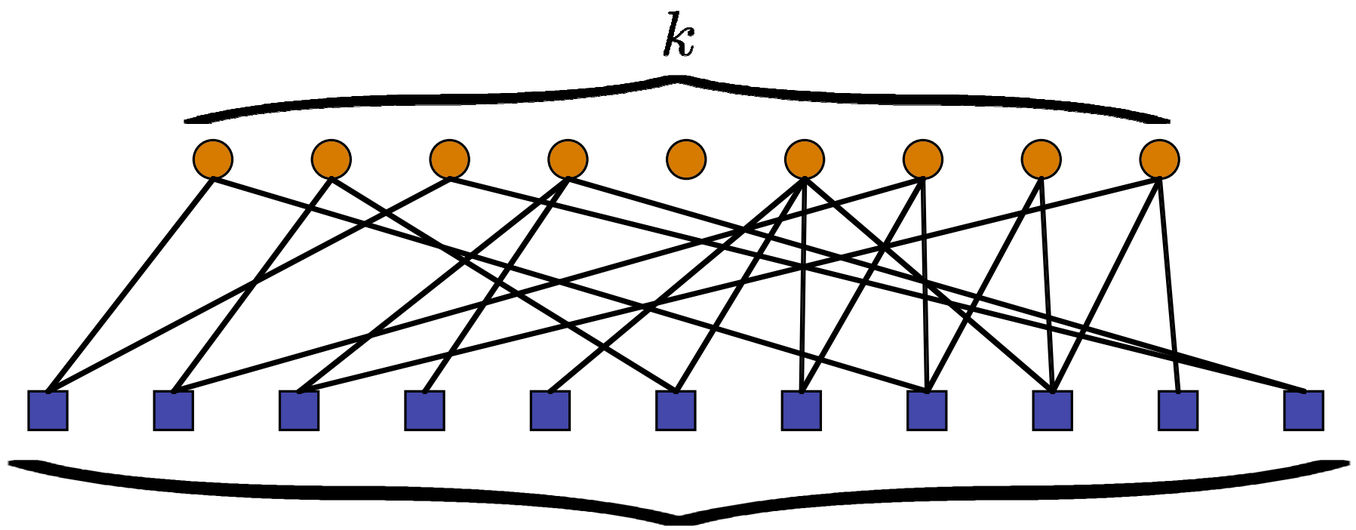
Expected weight of an output symbol is $\sum_d d\Omega_d = \Omega'(1)$.

Maximum likelihood decoding has cost $O(k^3)$.

Other methods (like Wiedemann's method) lead to a cost of $O(\Omega'(1)k^2)$.

But how about the overhead, with respect to the simple decoding algorithm given above?

Average Degree



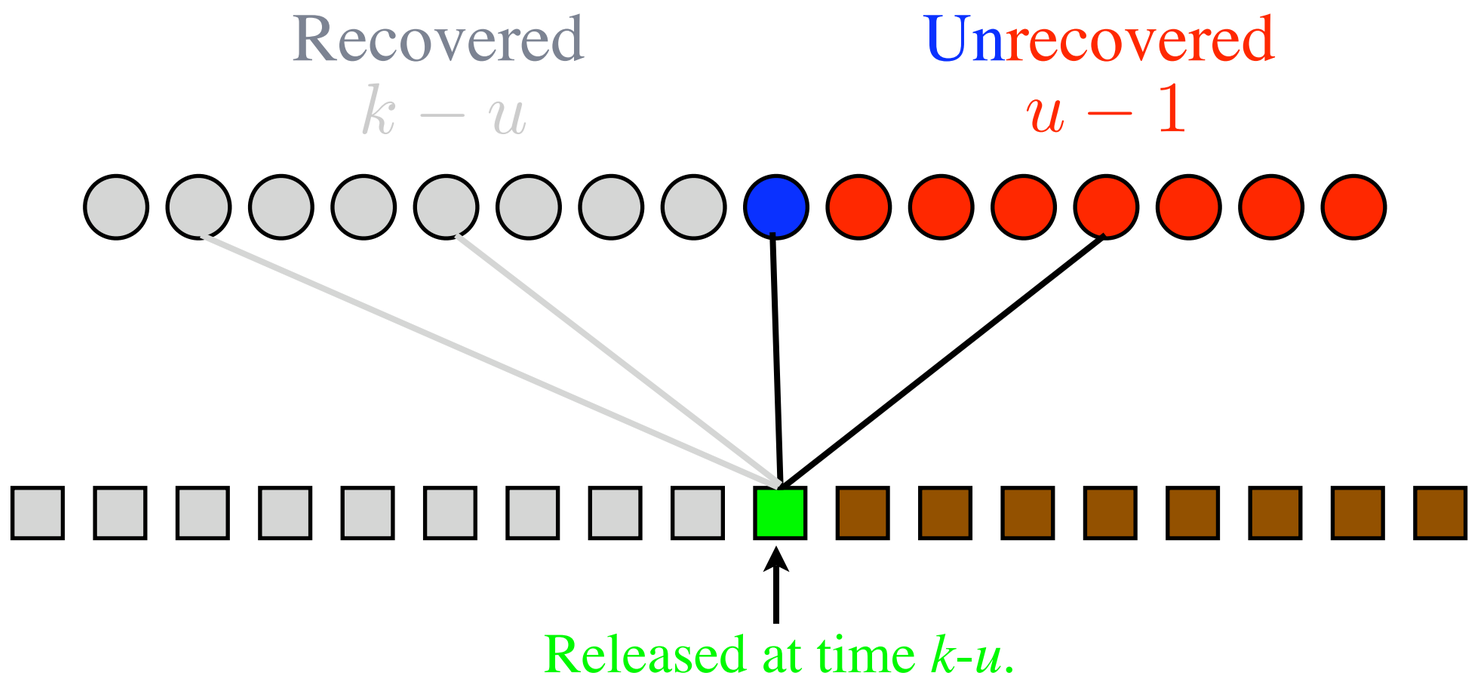
If an input symbol is not covered, then any decoding algorithm will fail.

What is the probability that an input symbol is not covered?

$$\left(1-\frac{\Omega'(1)}{k}\right)^{k(1+\varepsilon)} < \frac{1}{k^c} \Rightarrow \Omega'(1) > \frac{c}{1+\varepsilon}\ln(k)$$

Average degree must be at least logarithmic to ensure successful decoding.
Cannot guarantee constant encoding and linear decoding costs!

Release Probability



Probability that output symbol is released at time *k-u* is

$$\frac{1}{k}\sum_d \Omega_d \frac{\binom{k-u}{d-2}}{\binom{k-1}{d-2}} \frac{u-1}{k-d+1} d(d-1)$$

Release Probability

Exercise: $\sum_{d=2}^{k} \frac{\binom{k-u}{d-2}}{\binom{k-1}{d-2}} \frac{u-1}{k-d+1} = 1$ if $u > 1$.

Choosing

$$\Omega(x) = \frac{1}{k}x + \sum_{d=2}^{k} \frac{1}{d(d-1)}x^d$$

leads to one expected released output symbol per round! This distribution is called the Soliton distribution.

Its average degree is $\frac{1}{k} + \sum_{d=2}^{k} \frac{1}{d-1} \sim \ln(k)$, so has right order of magnitude.

But: fails miserably in practice, since we assume that on expectation only one symbol is released per round.

Robust Soliton

Luby has introduced a robust version of the Soliton distribution which works in practice.

The average degree of the distribution is $c\ln(k)$, where the constant c is directly related to the failure probability of the decoder.

The overhead of Luby's codes is $o(k)$, so the overhead decreases with the length.

This construction is absolutely remarkable in its simplicity, performance, and essential optimality!

However, we are still away from realizing our goal. For that, we need a new idea.

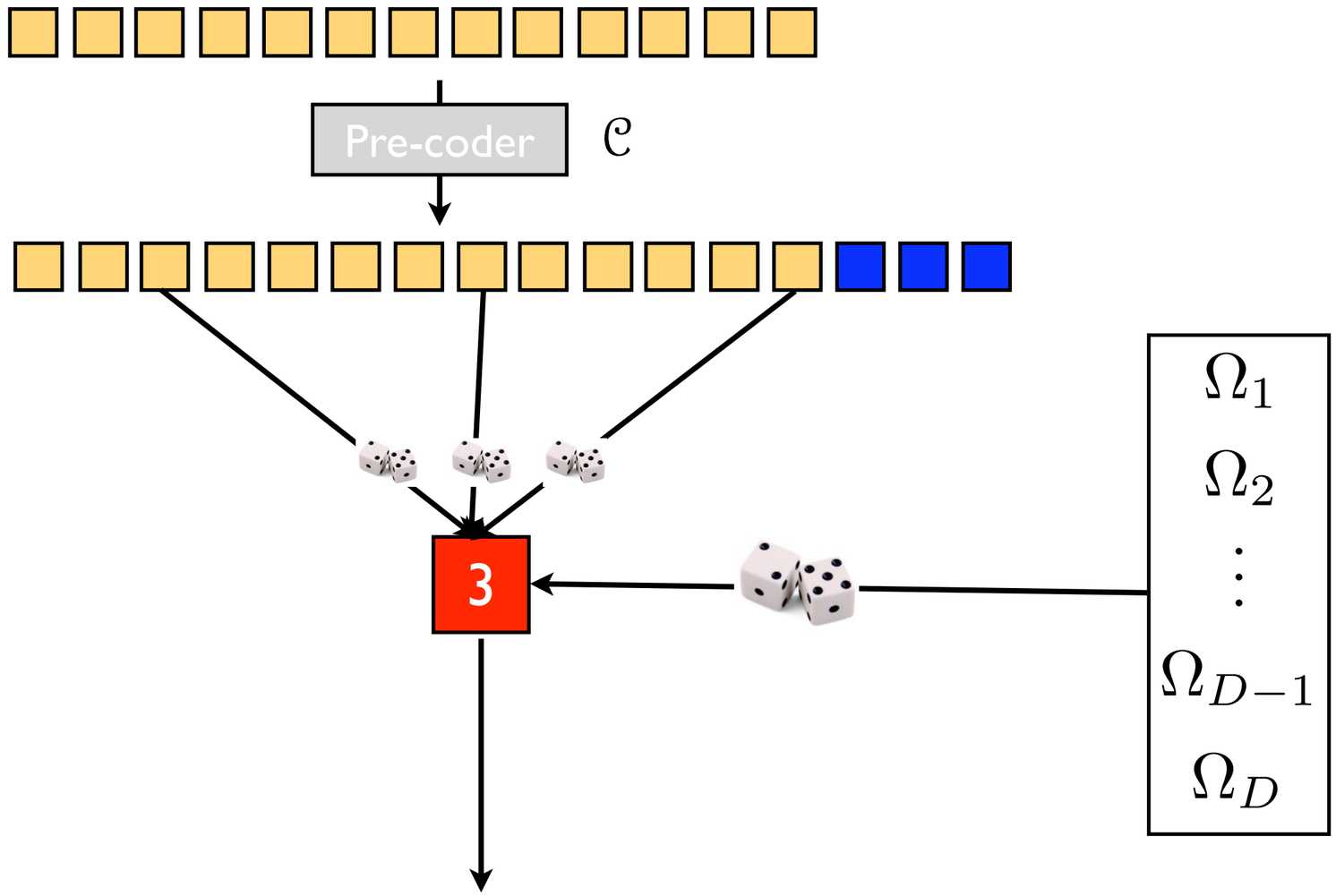
Average Degree

Constant average degree implies constant probability of error.

How can we achieve constant average degree and vanishing probability of error?

Raptor codes achieve this!

Raptor Codes



Parameters: $(k, \mathcal{C}, \Omega(x))$

Raptor codes are fountain codes (exercise).

Extremal Raptor Codes

Pre-code-only codes: $\Omega(x) = x$.

LT-codes: $\mathcal{C} = \mathbb{F}_2^k$.

It can be shown that for a pre-code of rate R the overhead of the pre-code-only code is at least -ln(1-R)/R, provided that the pre-code can be decoded perfectly.

Obtaining extremely small overheads means that the pre-code needs to be of very small rate. If the decoding algorithm of the pre-code is proportional to the length of the pre-code (true for LDPC codes), rather than the dimension, then the decoding cost of the pre-code scales with $1/R$, and so does the encoding cost.

Hence, pre-code-only codes are not great.

Systematic Raptor Codes

Suppose we already have a Raptor code with parameters $(k, \mathcal{C}, \Omega(x))$ which is very good with respect to some decoding algorithm. How can we construct a systematic code from this?

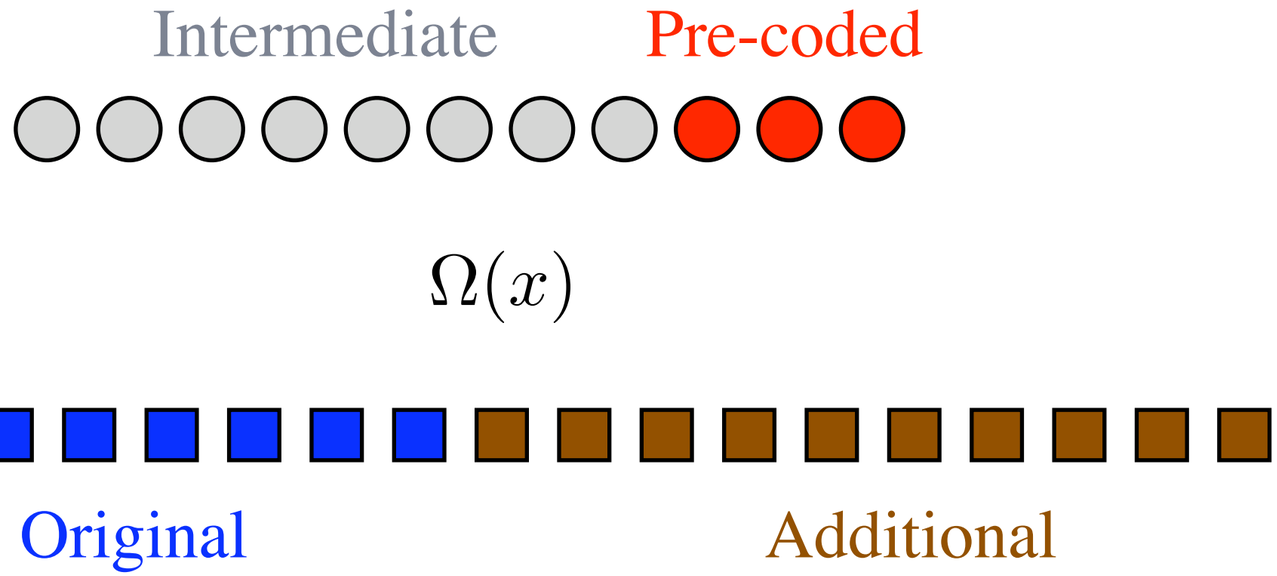
First idea: just send the original symbols, and then generate output symbols according to the normal encoding process.

This is a very bad idea. (Why?)

Second idea: suppose that we have identified k output symbols from which the original symbols can be decoded, and that the degree distribution of these output symbols is given by $\Omega(x)$.

Place the source into these k output symbols and decode to obtain k intermediate symbols. Generate the additional symbols from these intermediate symbols.

Systematic Raptor Codes



From the point of view of the intermediate symbols, the additional and the original symbols are equal, and generated according to the original Raptor code.

Since this Raptor code was good by assumption, the new code is truly systematic, and can be used in appliactions where such codes are required.

Path Forward

Raptor codes seem to be promising candidates for fountain codes with the properties we are interested in.

But to utilize them, we need to be able to design them properly.

This introductory lecture will not cover the design principles.

Structure of the Codes

Pre-code consists of two stages:

Source symbols get encoded using an LDPC code

Result of this encoding gets encoded using an HDPC code with fast encoding/ decoding algorithm.

The final result serves as input to an LT-code with weight polynomial

$$\Omega(x) = 0.0156\,x^{40}+0.0797\,x^{11}+0.111\,x^{10}+0.113\,x^{4}+0.210\,x^{3}+0.458\,x^{2}+0.00971\,x$$

http://www.3gpp.org/ftp/Specs/archive/26_series/26.346/26346-630.zip

Implementation and Performance

There are various implementations of these codes, ranging from open ones (due to BenQ (formerly Siemens)) do closed commercial implementations (by Digital Fountain).

The commercial implementations have been optimized to have minimal memory footprints and memory consumption.

They achieve impressive encoding and decoding speeds. For example, on a G5 processor, they run at around 3Gbps in software.

The error probability is also very small: for example, for k=1024, and 1% overhead, we achieve a block error probability if less than 1e-3. The probability decreases to 1e-6 when the overhead is 2%. In general, for quite some time, the error probability drops by a factor of 2 whenever a new symbol is received.

Other Standard Bodies

DVB-H: Raptor is selected for DVB-H IPDC file delivery.

IETF: Raptor for file delivery. Successfully past RMT working group last call.

3GPP2 (BCMCS): Raptor under consideration.

ATSC: Raptor under consideration.

DMB: Raptor under consideration.

DVB IPI: Focused on IPTV. Raptor under consideration.

VSF (Video Services Forum): Raptor under consideration.

And others....

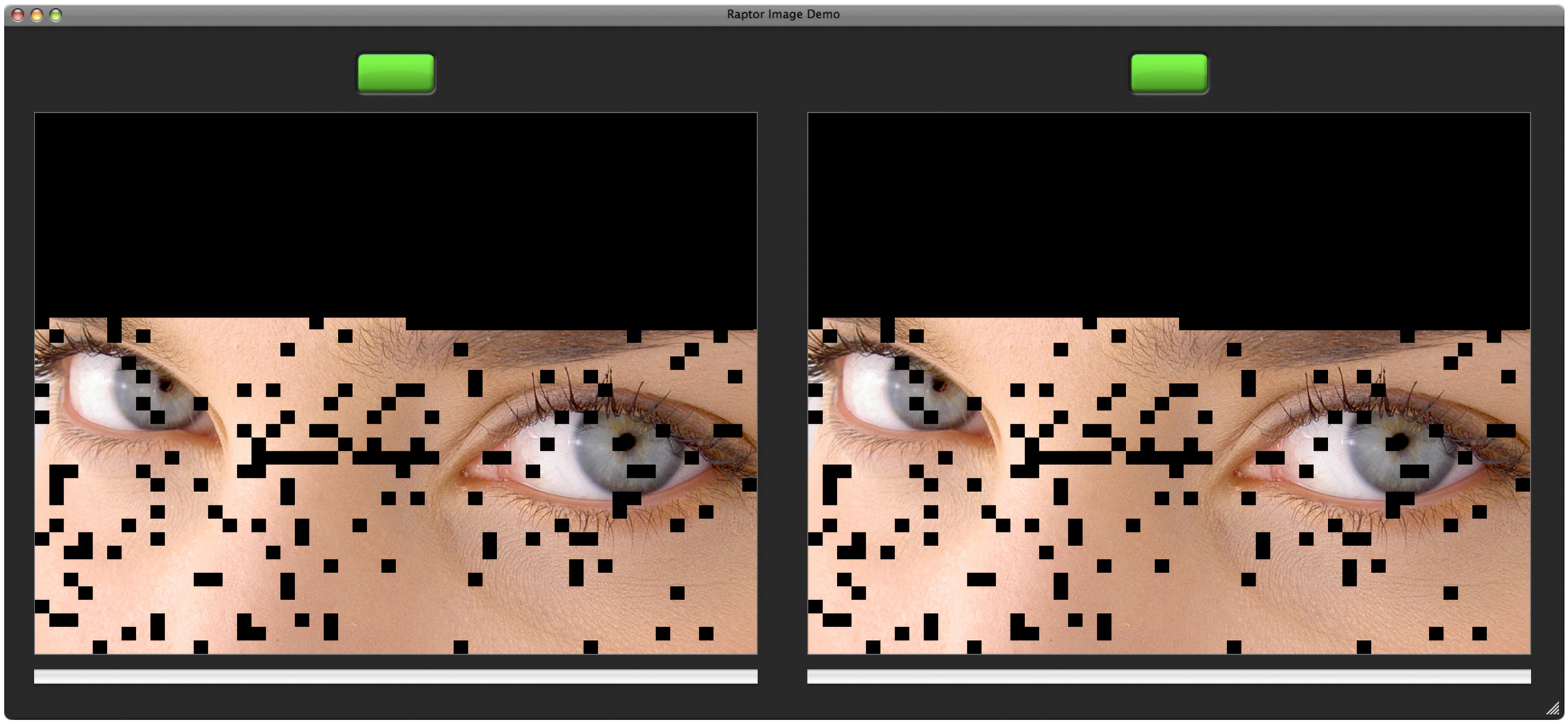
Demos

The coding technology used in the demos is the one standardized for 3GPP MBMS and DVB-H.

Demos were created by Dr. Giovanni Cangiani, and Mr. Zeno Crivelli (Dipl. EPFL), both members of the Laboratoire d'algorithmique.

Swisscom, January 2006

Demo 1: Single File Transmission

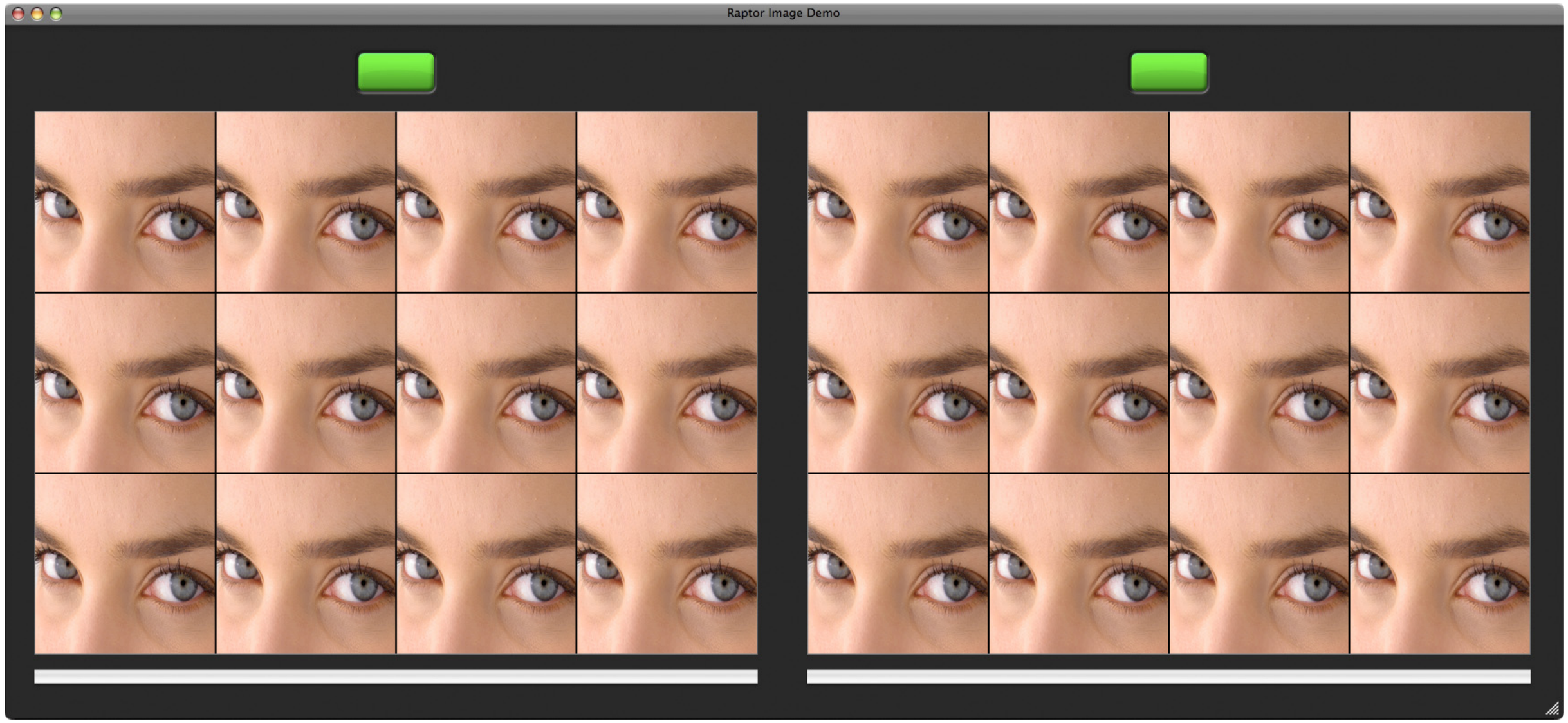


Carousel

Raptor

Swisscom, January 2006

Demo 2: Point to Multipoint Transmission



Carousel

Raptor

Demo 3: Streaming on 802.11

The demo uses three laptops, and one 802.11g access point.

One of the laptops is a server streaming two copies of a movie at 2Mbps eto ach.

Both are using udp connections.

One of the laptops shows the unprotected stream, while the other one is using the Raptor protected one.

(Some) References

1. J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data," Proc. of ACM SIGCOMM, 1998.

2. R. Karp, M. Luby, and A. Shokrollahi, "Finite Length Analysis of LT-Codes," Proc. of the International Symposium on Information Theory, 2004.

3. M. Luby, "LT-Codes," Proceedings of the 43rd Annual Symposium on Foundations of Computer Science, pp. 271-282, 2002.

4. A. Shokrollahi, S. Lassen, and R. Karp, "Systems and Processes for decoding chain reaction codes through inactivation," United States Patent, Serial Number 6,856,263.

5. A. Shokrollahi and M. Luby, "Systematic encoding and decoding of chain reaction codes," Published patent application, Serial number 20040075593.

6. A. Shokrollahi, "Raptor Codes," to be published in the IEEE Trans. Inf. Theory, 2006.