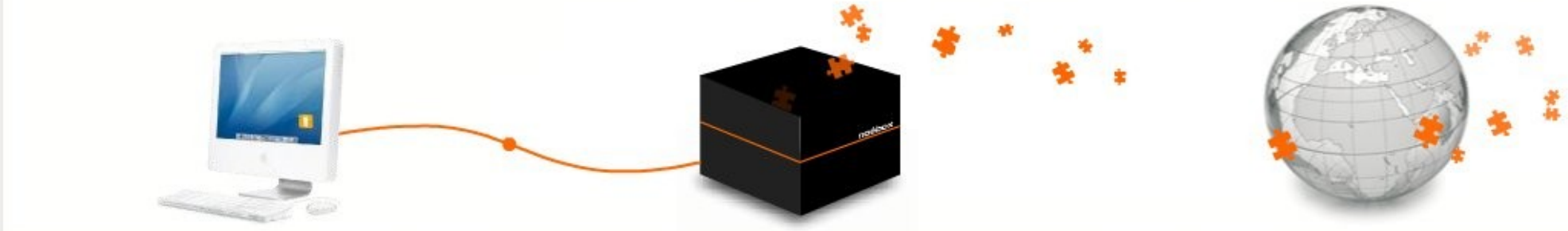
Using P2P
in backup and storage systems

Sébastien Choplin
(Nice-Sophia M2/PhD Alumni)



Agenda

- Generalities about P2P storage
- Redundancy
- Confidentiality
- Connectivity
- Localisation
- Existing Solutions

Introduction

- P2P: Sharing Ressources
- File Publishing: File (read only)
  - Napster, Gnuttella, Freenet, eMule, bittorrent...
  - Goal: sh! (MP3, divx ;-)
- Distribute Computation: CPU
  - Seti@Home, XtremWeb, ...
  - Goal: Huge Computation
- Distribute Storage: Disks
  - InterMemory, PAST, OceanStore, CFS, IVY, ...
  - Goal: Long Term Storage

What is P2P ??

(slides Franck Cappello JRES 2005)

- Def 1: "A class of applications that take advantage of resources (e.g., storage, cycles, content) available at the edge of the Internet." ('00)
  - Edges often turned off, without permanent IP addresses, etc.
- Def 2: "A class of decentralized, self-organizing distributed systems, in which all or most communication is symmetric." (IPTPS'02)
- many others..., no consensus
- Each node:
  - can be client as well as server
  - can communicate directly
  - can be involved in the infrastructure maintenance

P2P storage

Which properties should such a system have ?

- Publishing vs Storing
  - publishing:
    - availability for many peers
    - perennity can depend of popularity
    - access control or not
  - storing:
    - availability for few peers
    - perennity can depend of data life-cycle
    - access control required

P2P storage

- Data life-cycle
  - for how long time should you maintain perennity ?
- Data write access
  - should data be modified ?
  - should it be modified by more than 1 peer ?
- Data read access
  - atomic ? partial (streaming) ?
- Which kinds of peer ?
  - Personal computer
  - Computers with high availability
- Do you know something usefull about peers ?

UbiStorage

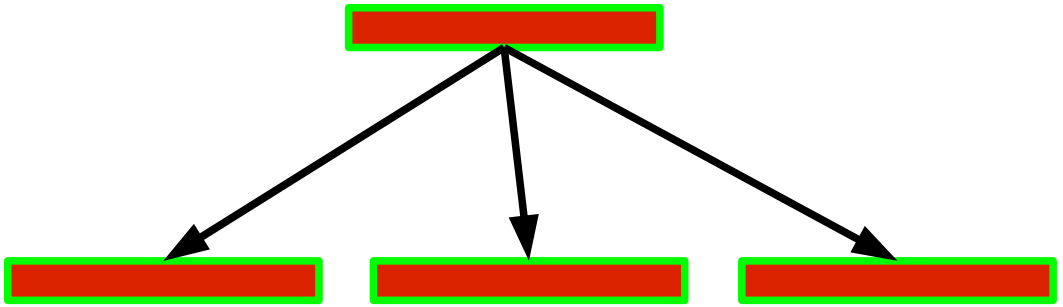
Redundancy

How to Make Permanent Data?

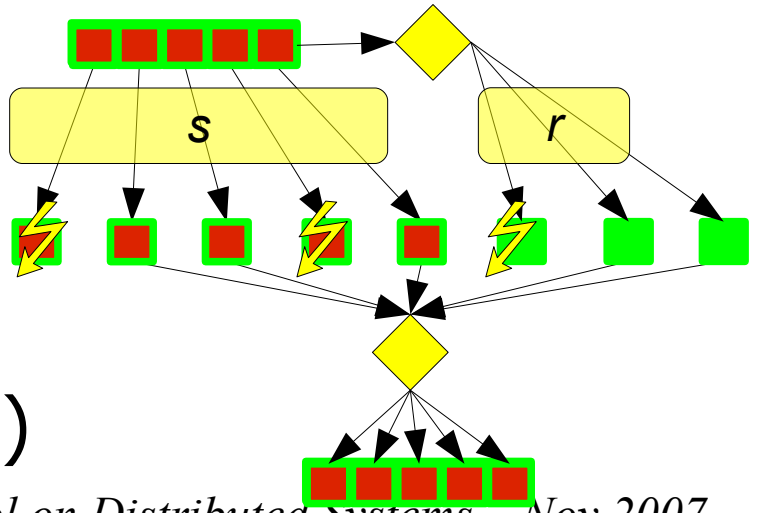
- Facts:
  - Peers are volatile
  - Peers are not always available
- Redundancy
  - Minimize
    Useful Space ratio = dataSize / dataConsumed
- Self Repair Mechanism
  - Fault Detection
  - Reconstruction

Redundancy

- Replication:
  - Data Duplicated k times
  - Tolerancy: k -1 failures
  - Useful Space: 1/k

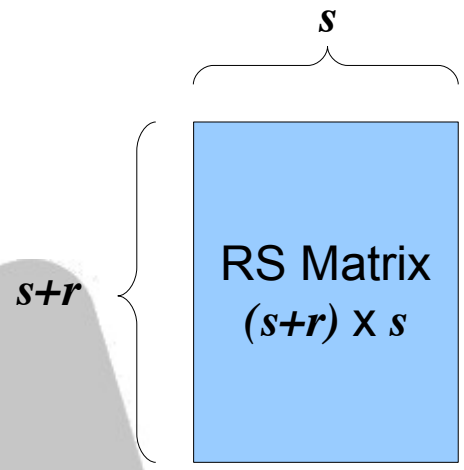


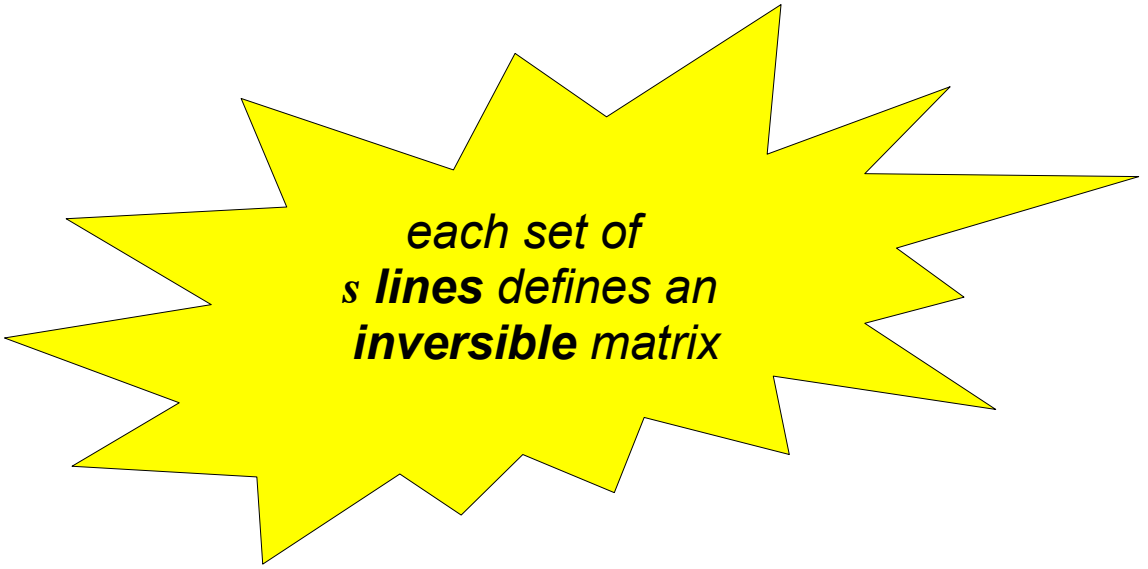
- IDA Algorithm:
  Erasure Codes (Reed Solomon)
  - Data fragmented: s
  - Redundancy: r
  - Tolerancy: r failures
  - Useful Space: s / (s+r)
- Exemple : k = 2 vs (s,r)=(16,16)



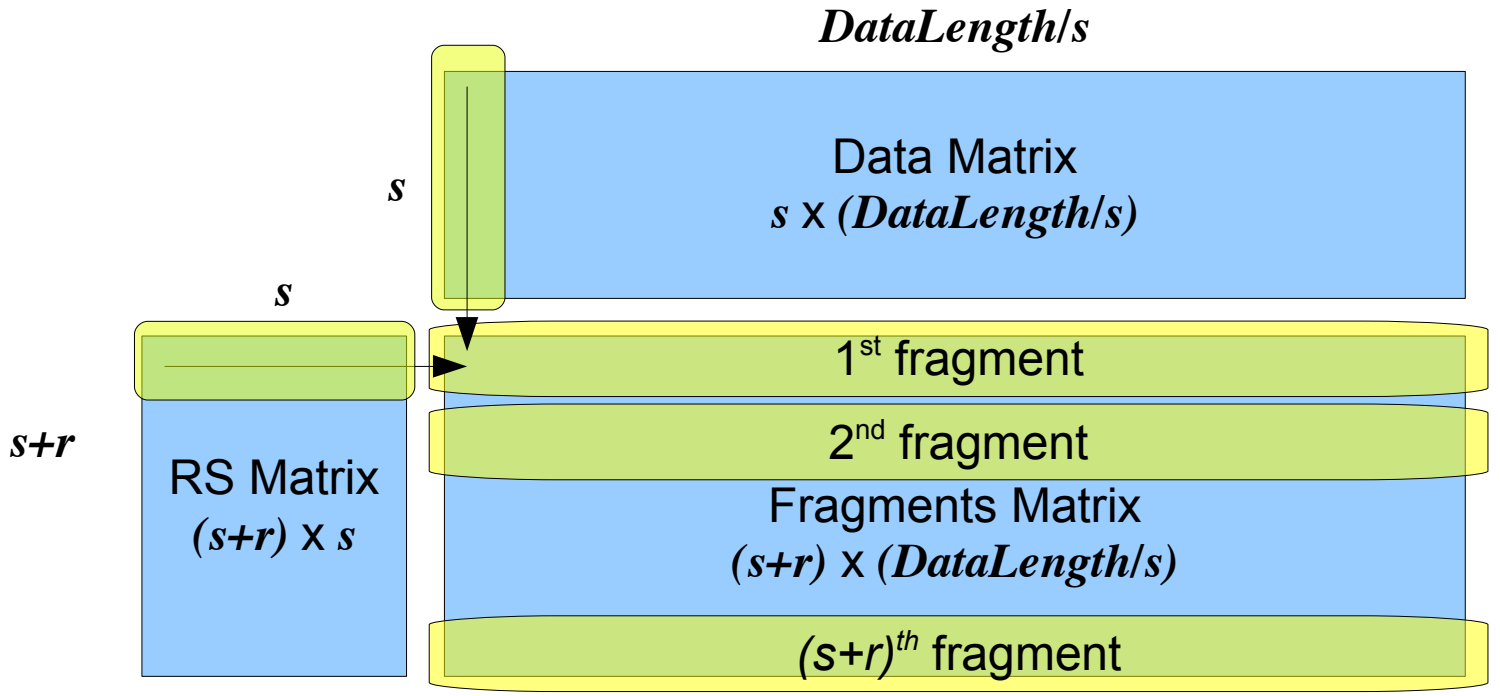
Reed Solomon : Theory

- Based on erasure codes (Galois theory)
- «A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems » (James S. Plank) for implementation
- Given s,r : generation of a coding Matrix over Galois Field GF(2^n)



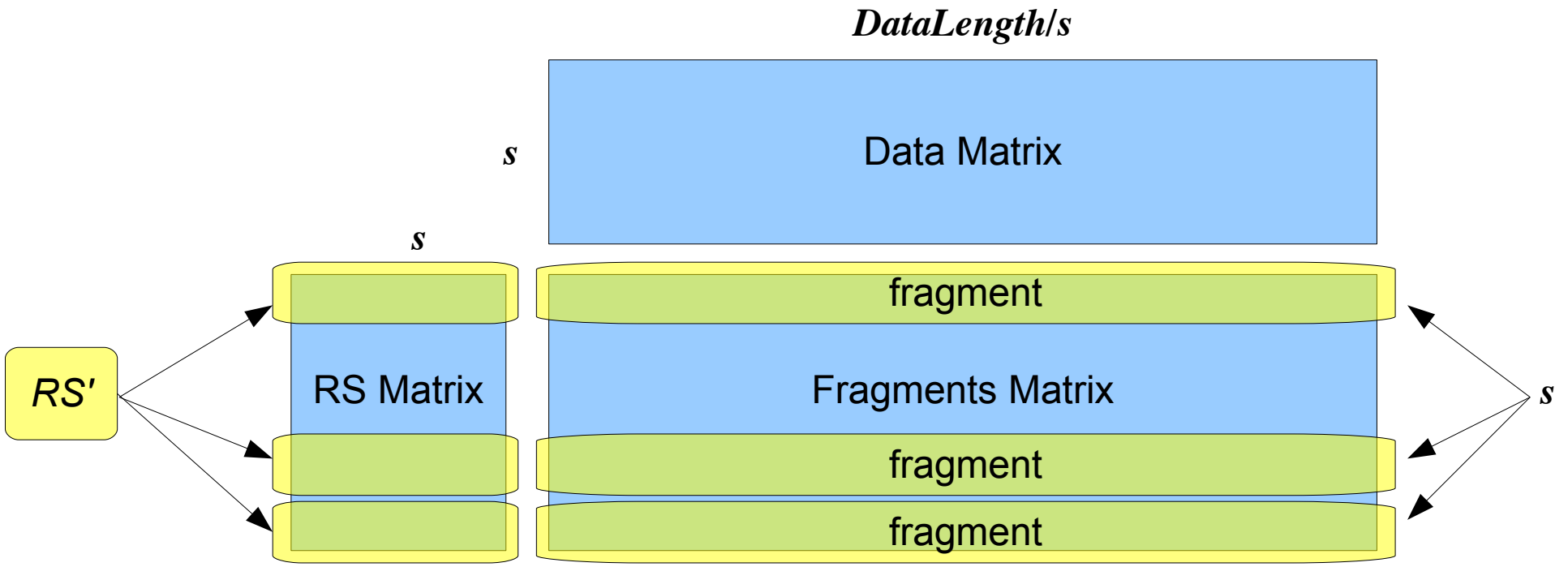


Fragments generation



$$RSMatrix * DataMatrix = FragmentsMatrix$$

Data regeneration



***RS'**: s x s*
inversible matrix

RS' x DataM = FragM'

DataM = inv(RS') x FragM'

Reed Solomon : example

- Example with GF(4): {0,1,A,B}
- s=3, r=1

neutral element

+	0	1	A	B
0	0	1	A	B
1	1	0	B	A
A	A	B	0	1
B	B	A	1	0

neutral element

x	0	1	A	B
0	0	0	0	0
1	0	1	A	B
A	0	A	B	1
B	0	B	1	A

Id

$$RS = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

V. d. Monde

$$* \quad \begin{pmatrix} 0 & \cdots \\ 1 & \cdots \\ A & \cdots \end{pmatrix}$$

Data matrix

$$\begin{pmatrix} 1 \\ A \\ B \end{pmatrix}$$

that's all

Coding matrix

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & \cdots \\ 1 & \cdots \\ A & \cdots \\ B & \cdots \end{pmatrix}$$

Fragments matrix

invertion

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ A \end{pmatrix}$$

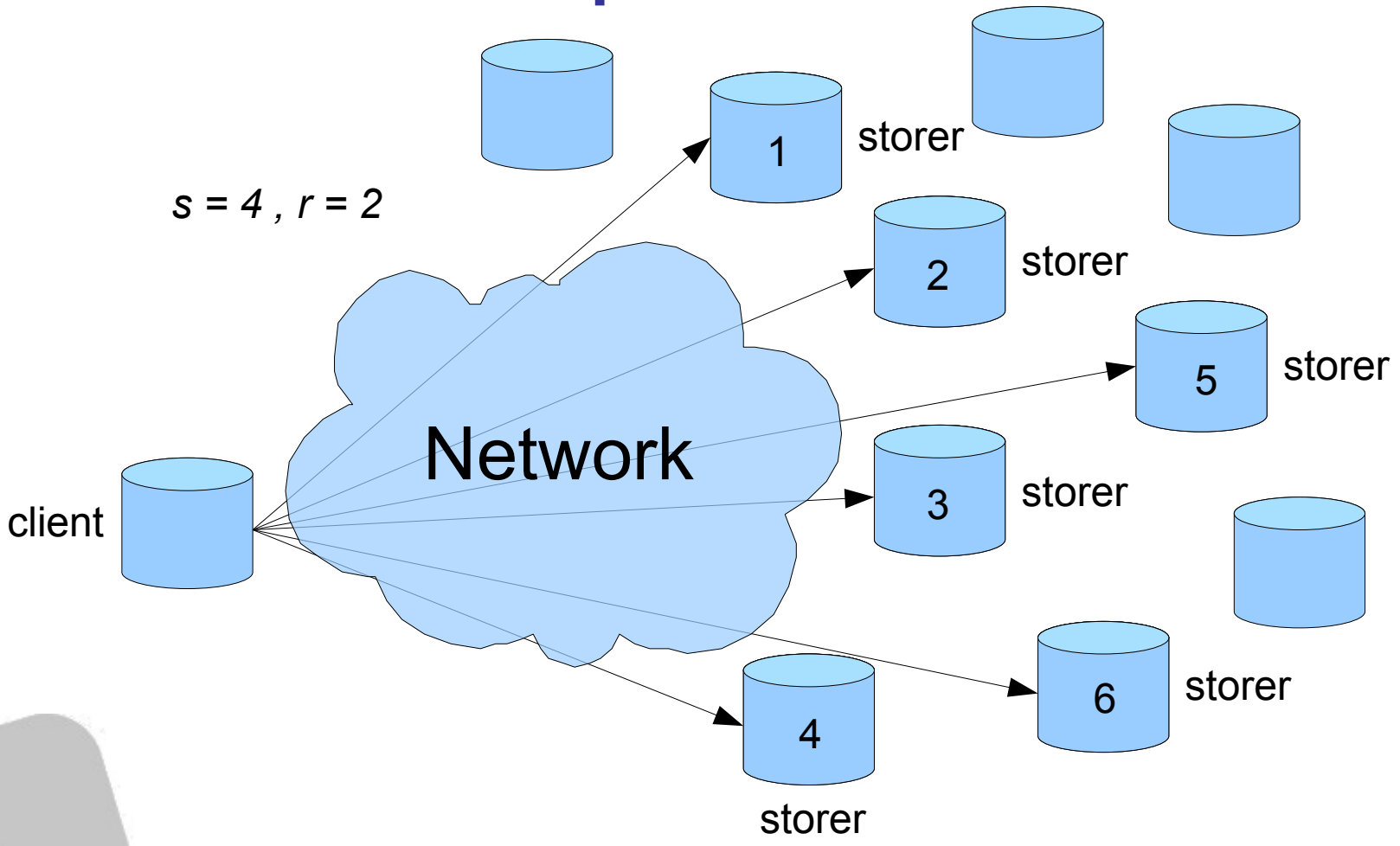
In practice: use GF(256) each elements represents a byte

Reed Solomon P2P operations

Unitary Operations :

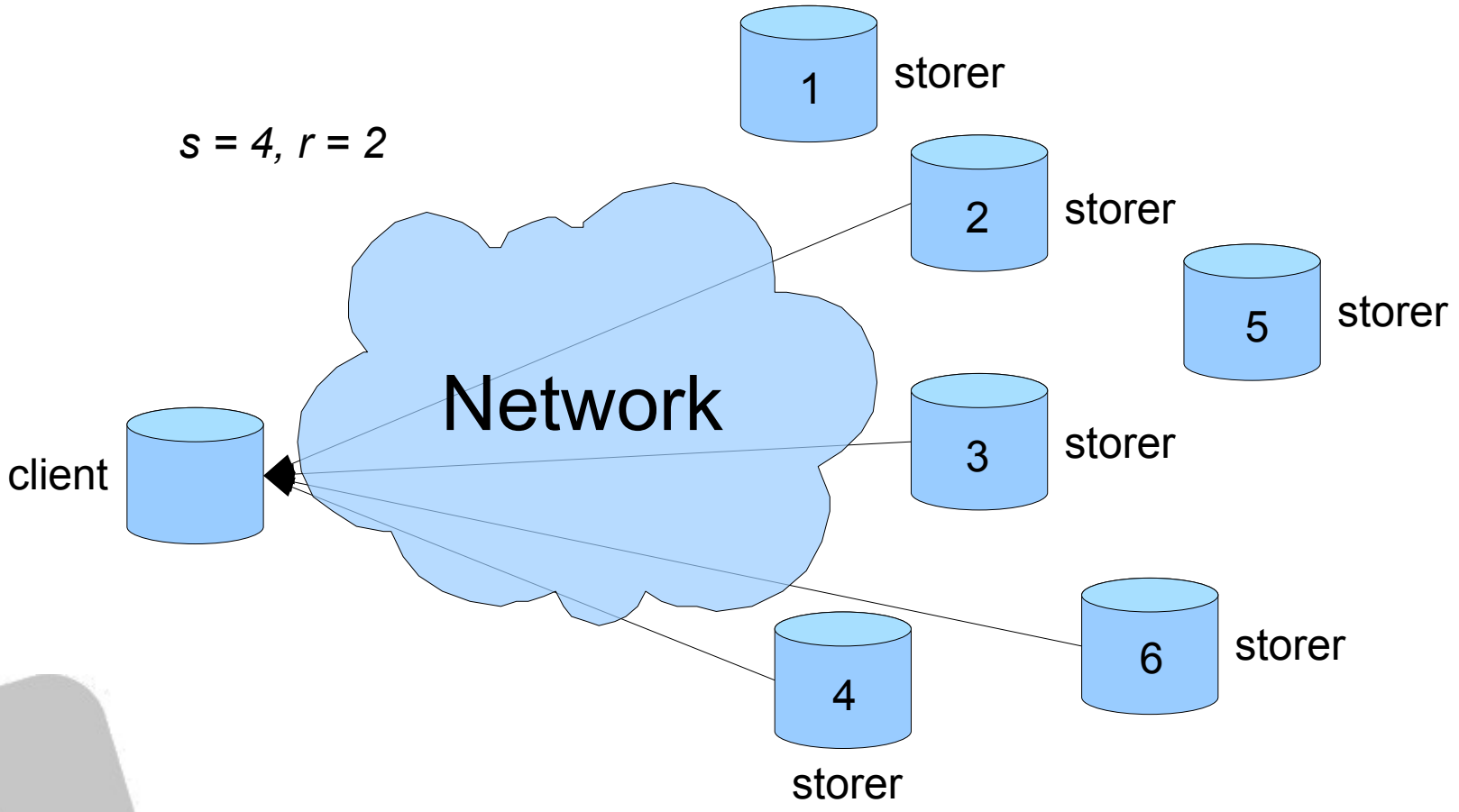
- put (inject data)
- get (retrieve data)
- rebuild (reconstruct missing fragments)
- free (delete data)

Reed Solomon : put



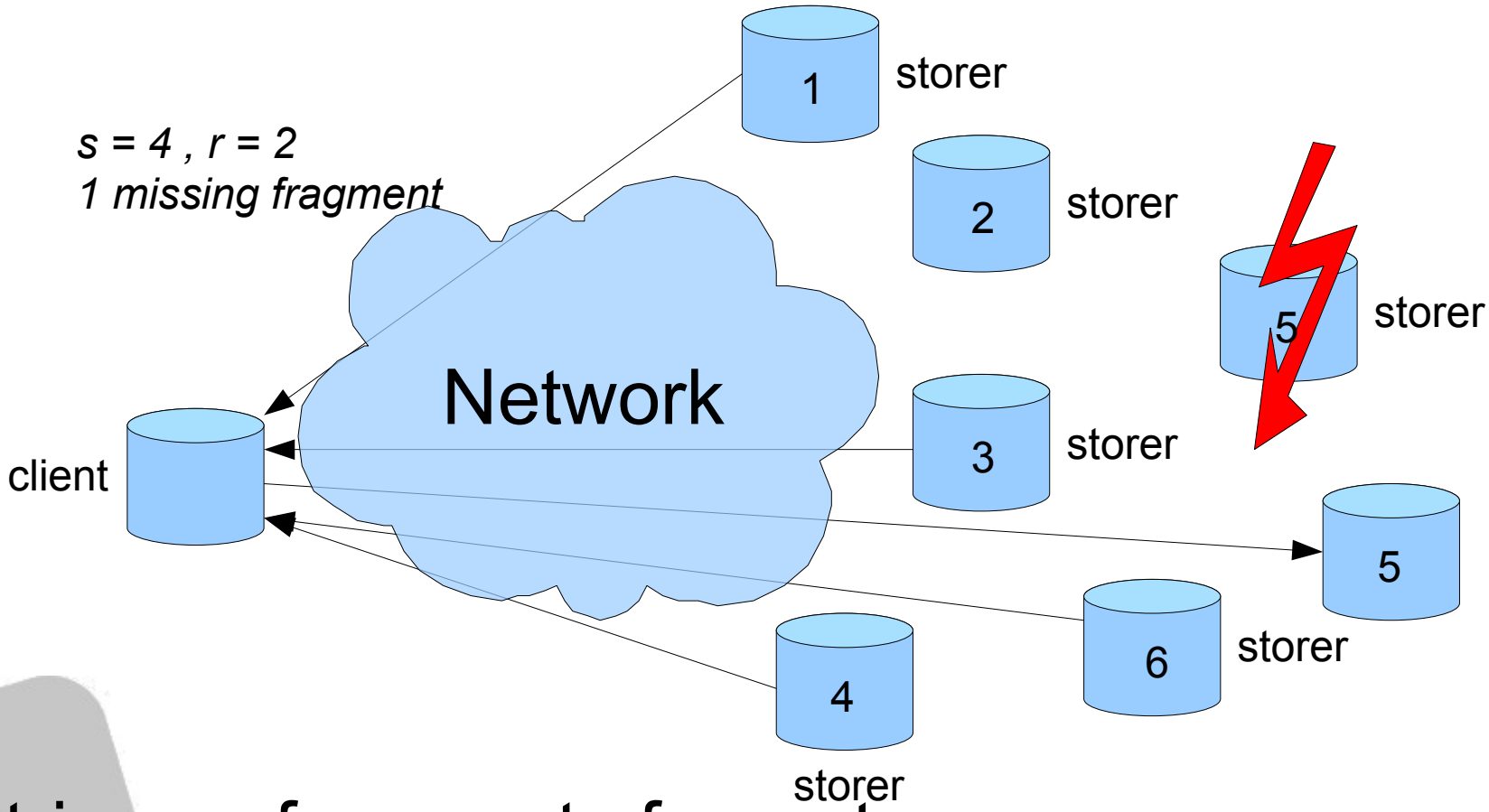
Send *s+r* fragments to storer peers

Reed Solomon : get



Retrieve *s* fragments from storer peers

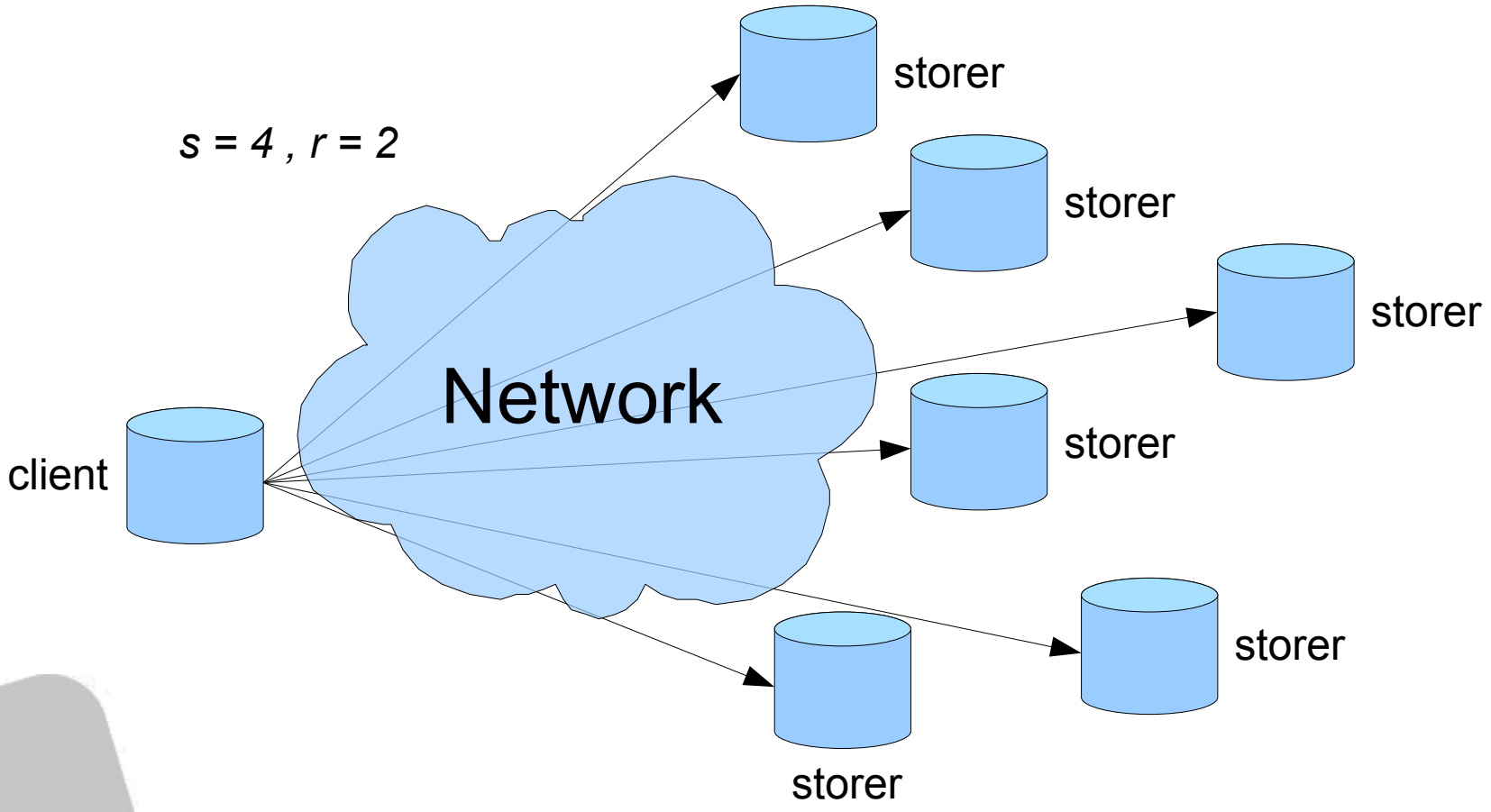
Reed Solomon : rebuild



Retrieve *s* fragments from storers

Send rebuilded fragments to other storers

Reed Solomon : free



Say *s*+*r* storers to delete fragments

Reed Solomon costs

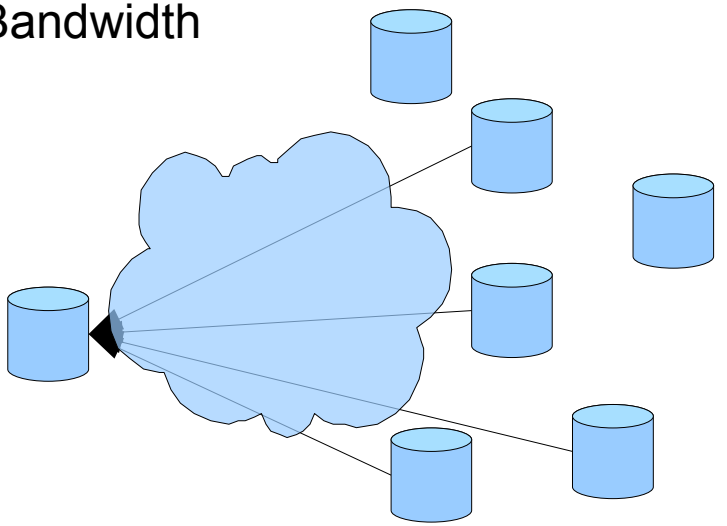
- Objectives:
  - operation completion ?
  - set of operations completion ?
- What should be optimized ?
  - delay of 1 unitary operation ?
  - delay of a set of unitary operations ?
  - amount of transferred bytes ?
  - do you consider homogenous peers set ?

Reed Solomon costs hypothesis example

- CPU time << Transmission time
- All peers have the same role (client/storer)
- All peers have almost the same symetric bandwith ?
  - depends of country, in France ratio of 10 between upload and download
  - bandwith is inexpensive, but depends of country
- Most of the time, peers' bandwith is not used
- Main objective is to reduce delays of operations

Reed Solomon delay : get

s, r, fragmentSize, DownloadBandwidth, UploadBandwidth

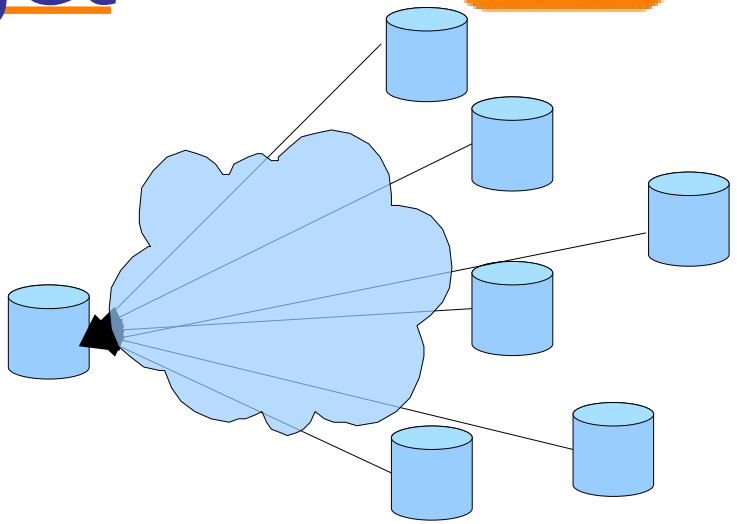


- Optimum delay is

$$\frac{s * fragmentSize}{DownloadBw}$$

- get s fragments from s storers
  - $UploadBw*s < DownloadBw$
    - bottleneck is storers upload bandwidth
    - if one storer is overloaded : the operation is delayed by this storer
      why not getting from all storers ?

Reed Solomon delay : get



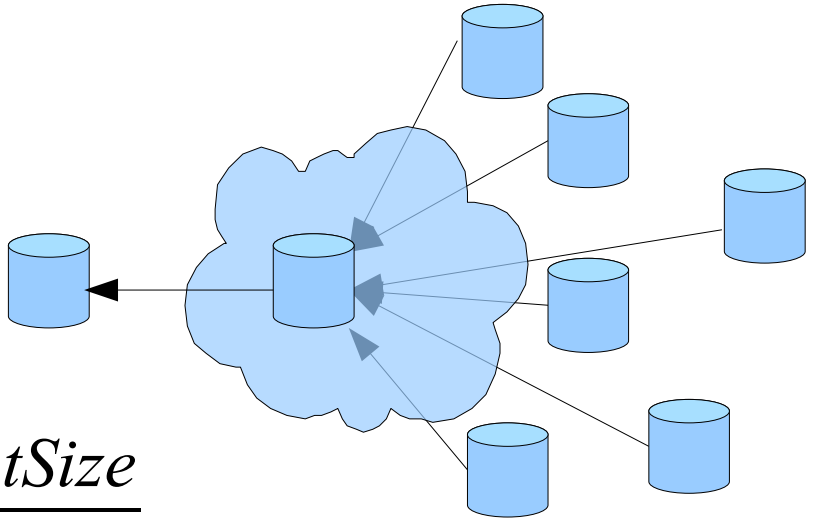
get ***s+r*** fragments from
s+***r*** storers

- ***UploadBw*s < DownloadBw***
  - get fragments from faster storers, seems ok
- ***UploadBw*s > DownloadBw***
  - bottleneck is client download bandwidth: useless delayed. Bbytes will be discarded, worse case if all peer have same bandwith: discard ***r*** full downloaded fragments

$$delay = \frac{(s+r)*fragmentSize}{DownloadBw} \leqslant optimum + \frac{r*fragmentSize}{DownloadBw}$$

Reed Solomon delay : get

Get help from high bandwith peer ?



$$delay \leqslant optimum + \frac{fragmentSize}{UploadBw}$$

Streaming in high bandwith peer

$$delay = optimum$$

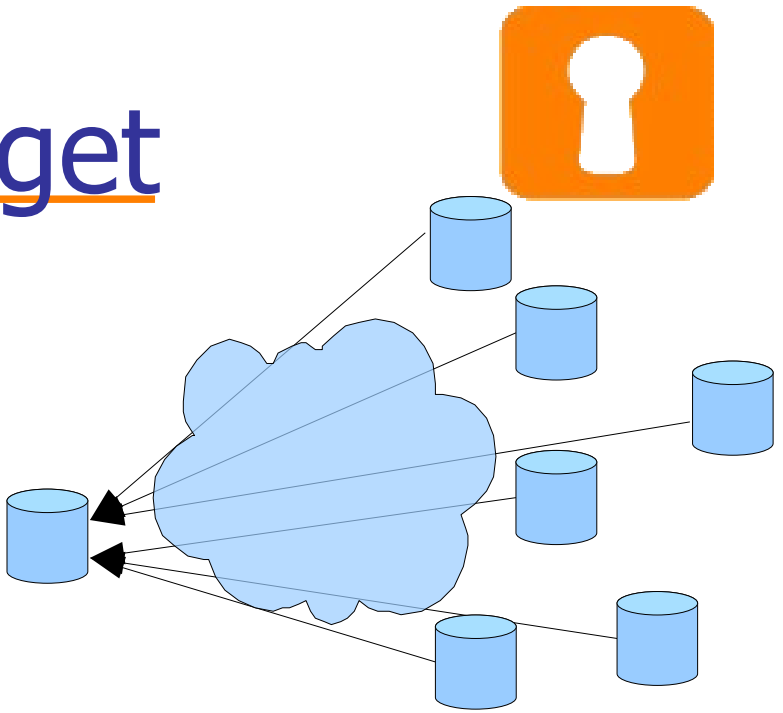
StorerUploadRate = s^{th} worse upload rate

useless use of slowest storers

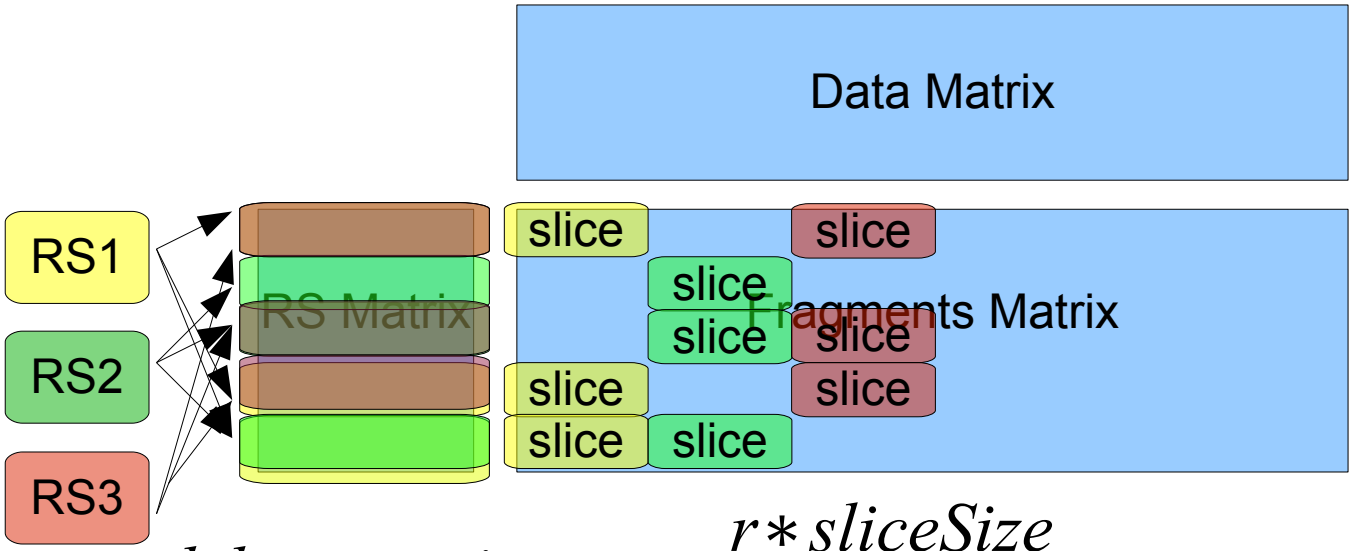
scalability

Reed Solomon delay : get

Reed Solomon uses
Linear Operations



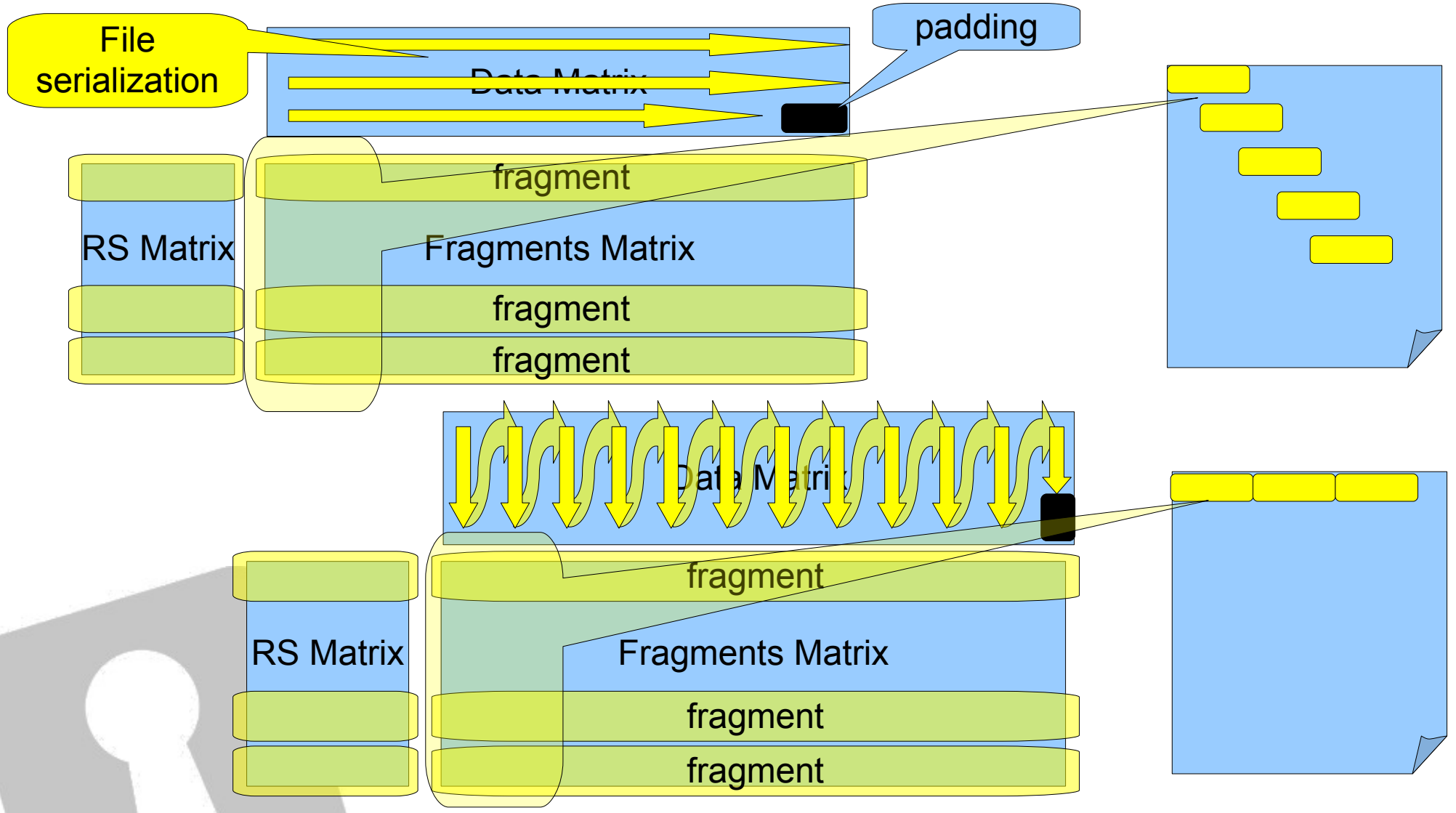
slice fragments and
operate on slices



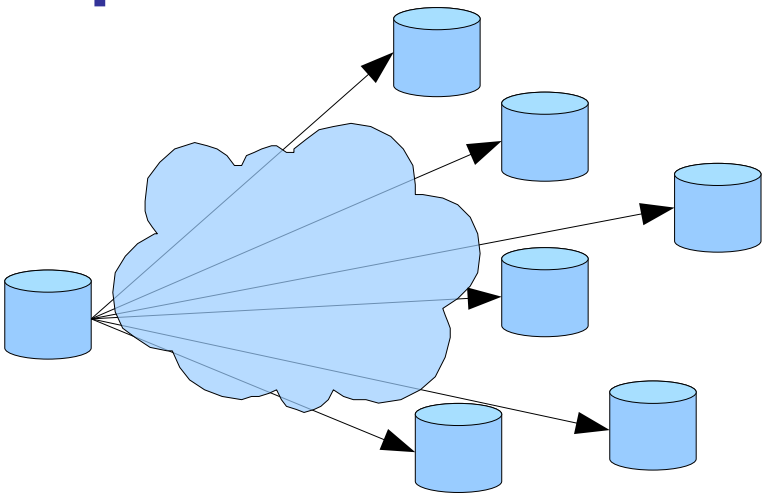
get from available
storer next usefull slice

$$delay \leqslant optimum + \frac{r * sliceSize}{DownloadBw}$$

File serialization, streaming



Reed Solomon delay : put



- Optimal delay is

$$\geqslant \frac{s * fragmentSize}{UploadBw}$$

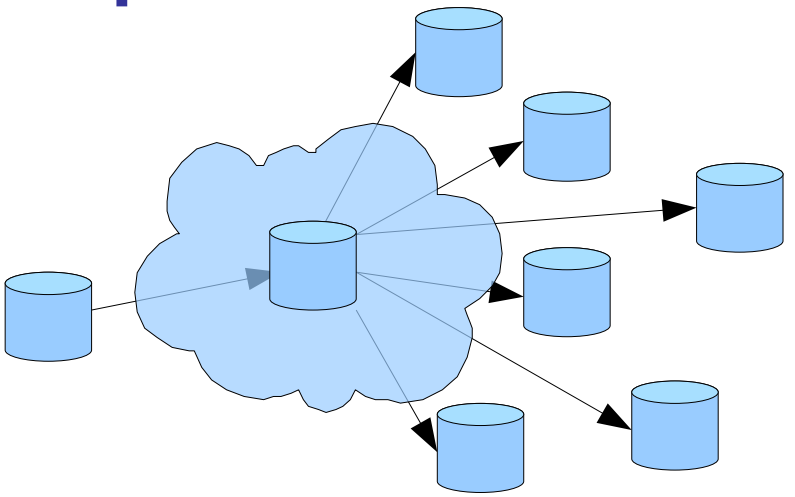
- send s+r fragments to s+r peers
  - bottleneck: sender peer bandwith
    (if other peers are not overloaded)

$$delay \leqslant \frac{(s+r) * fragmentSize}{UploadBw}$$

  - how to reduce this cost ?
    can a peer send only s fragments and delegate the construction of the r remaining fragments to others peers to speed up the process ?

Reed Solomon delay : put

Get help from high bandwith peer ?

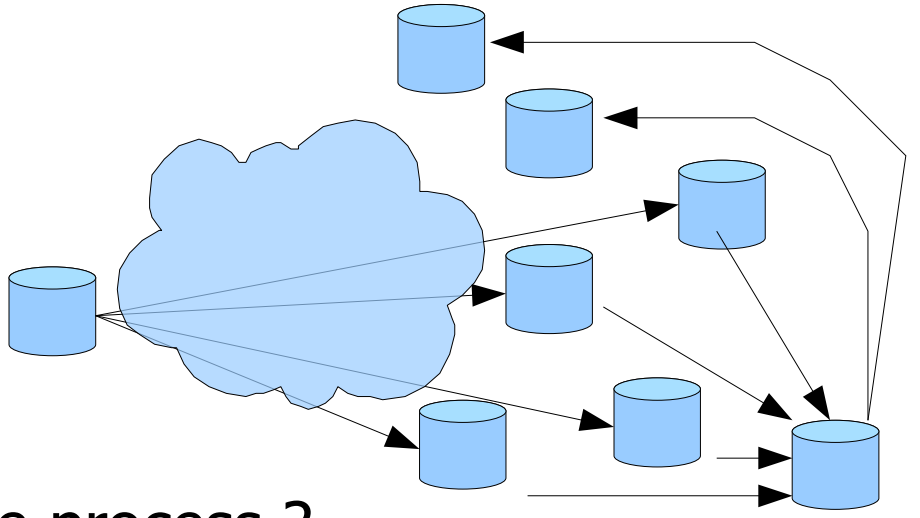


$$delay \leqslant optimum + \frac{fragmentSize}{DownloadBw}$$

StorerDownloadRate = worse download rate

but 1 overloaded storer will delay the process

Reed Solomon delay : put



can a peer send only s fragments and delegate the construction of the r remaining fragments to others peers to speed up the process ?

- no streaming:

$$delay \leqslant optimum + GetCost + \frac{r * fragmentSize}{UploadBw}$$

- streaming:

$$delay \leqslant optimum + \frac{r * sliceSize}{UploadBw}$$

-

send only s+k fragments ? hidden cost ?

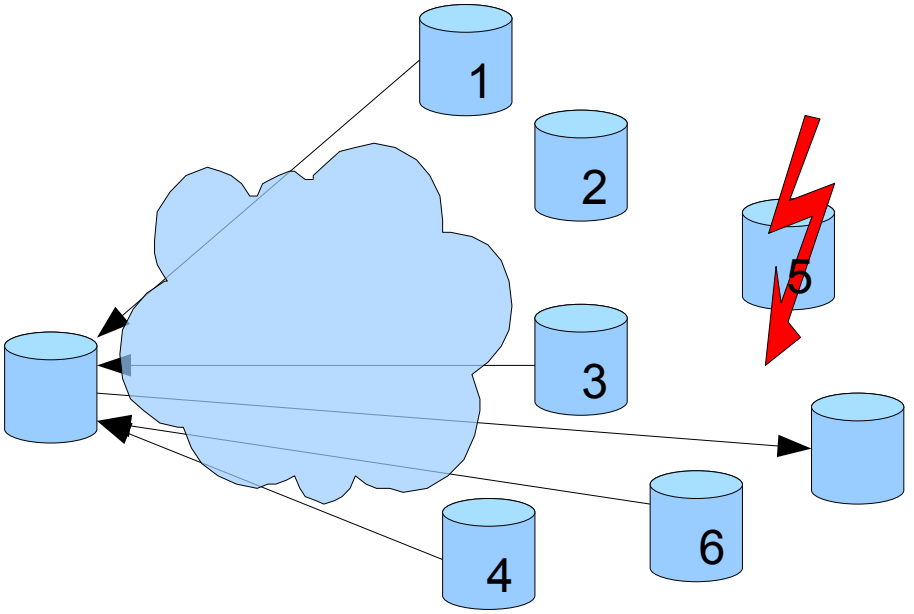
Reed Solomon delay : free

- No data transfer, only meta-information

$$delay = O(1)$$

But garbage collection mechanisms for disconnected peers ...

Reed Solomon delay : rebuild



For replication:
only 1 peer is required
to rebuild a replica

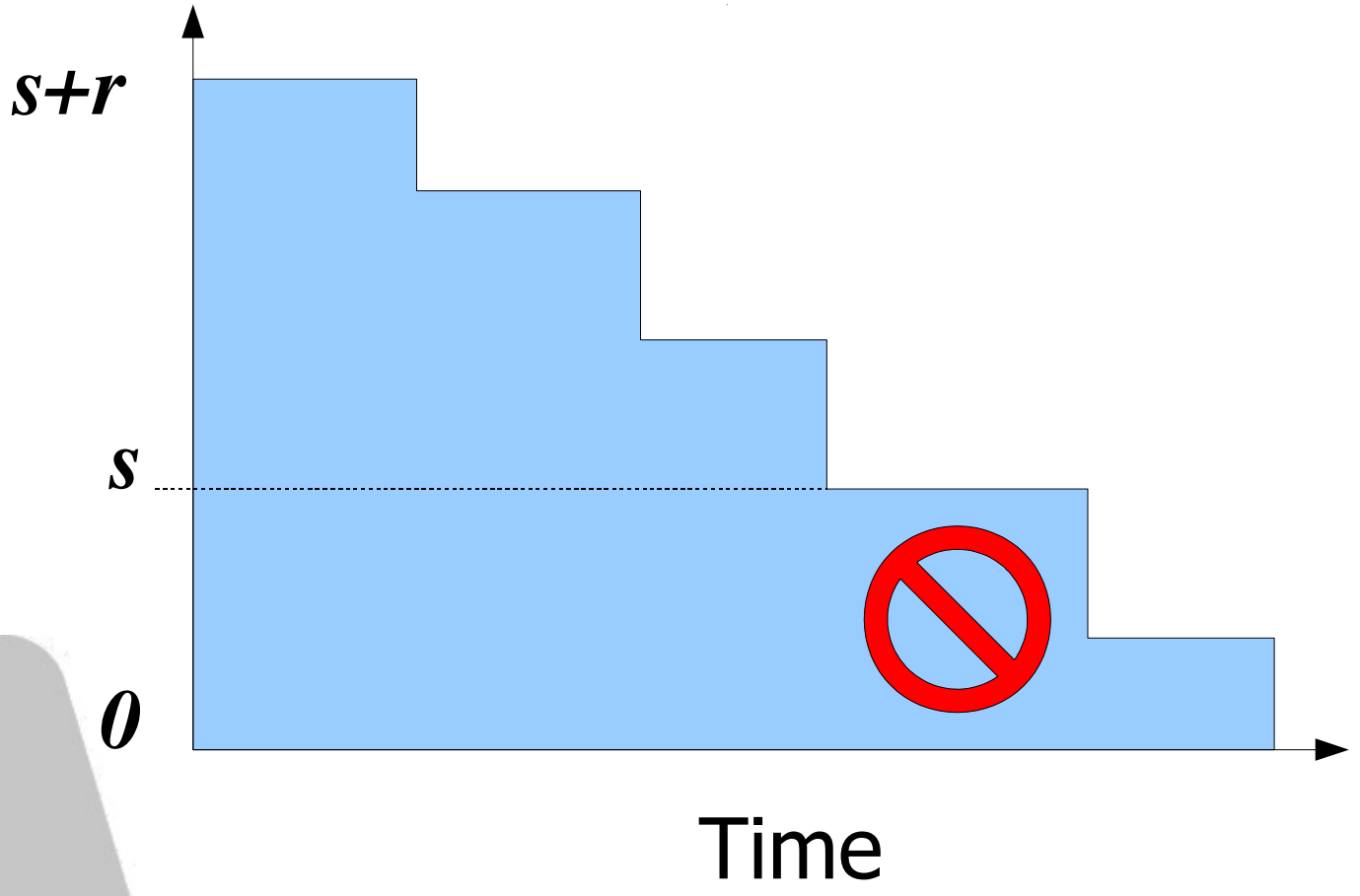
For Reed Solomon:
s peers are necessary
to rebuild one fragment

- cost of a get + cost of sending rebuilded frags

- Rebuilding in RS scheme is expensive
- When rebuild ?

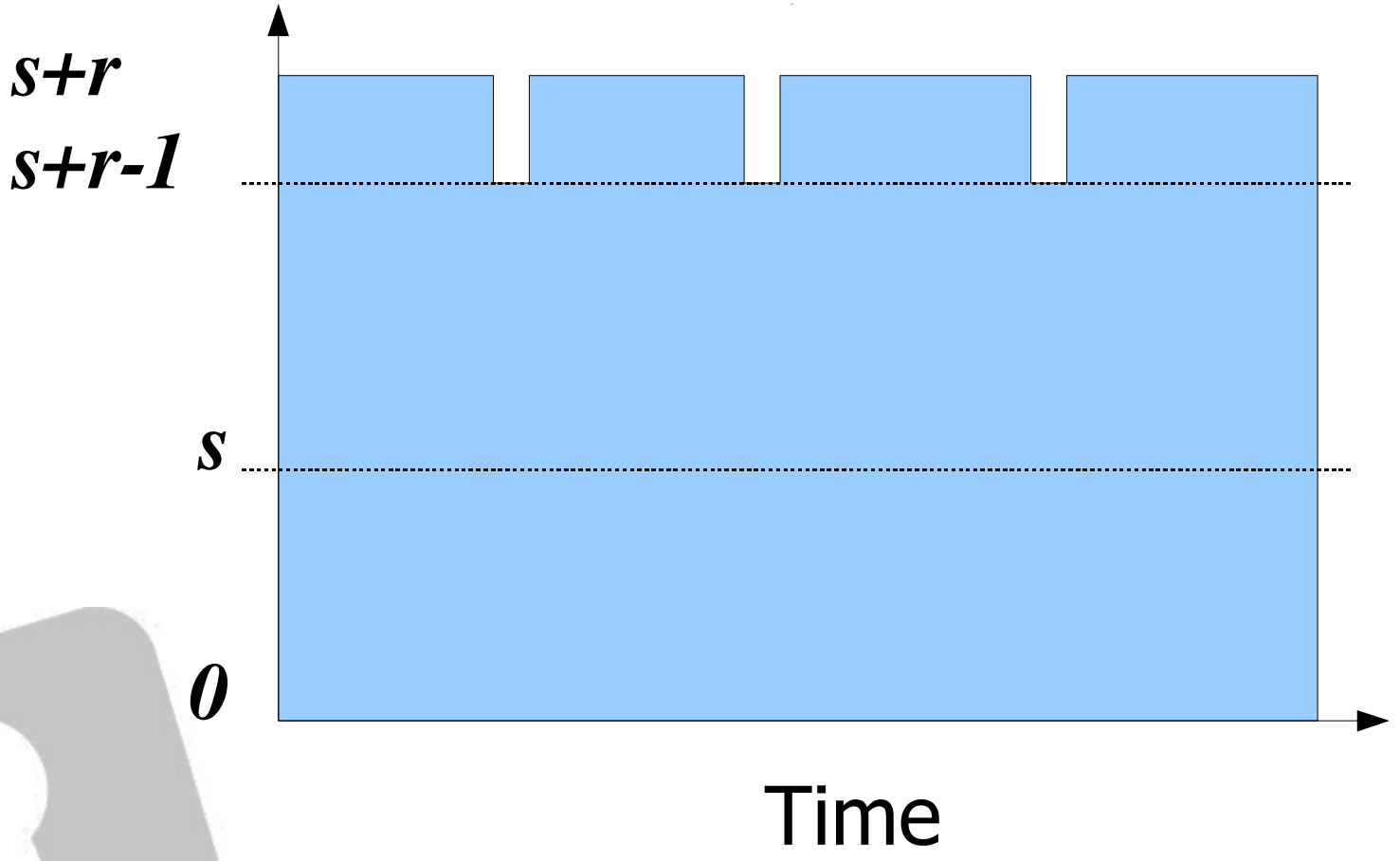
Reconstruction strategy

- Don't reconstruct at all:



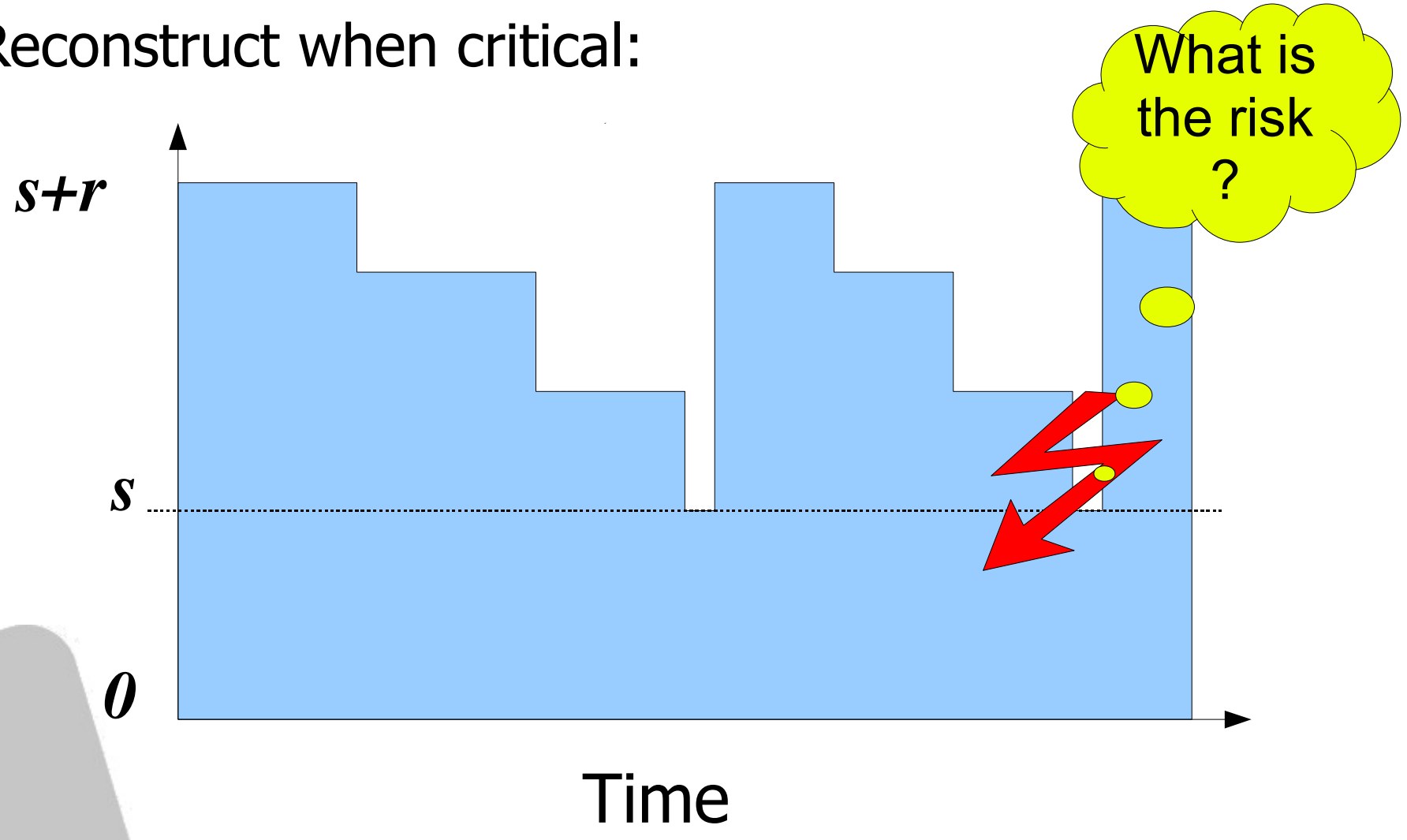
Reconstruction strategy

- Reconstruct ASAP:



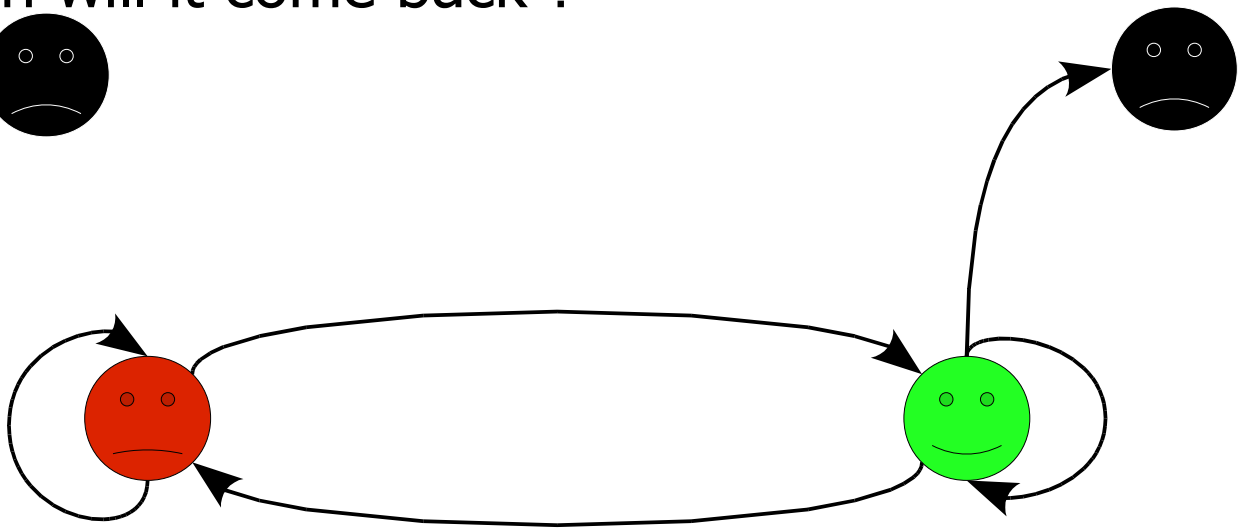
Reconstruction strategy

- Reconstruct when critical:



Peer behaviour

-
- Available
  - for how long ?
- Disconnected
  - when will it come back ?
- Died



Exponential distribution

- Model the behavior of units that have a constant failure rate.
  - An exponential distribution can easily be described as follows...:
    If a thousand items have a constant failure rate of 10% / month.
    After the 1th month 100 items have failed (0.1x1000) leaving 900.
    After the 2nd month 90 items will have failed (0.1x900) leaving 810.
    After 12 months 31 items will fail leaving 282 items.

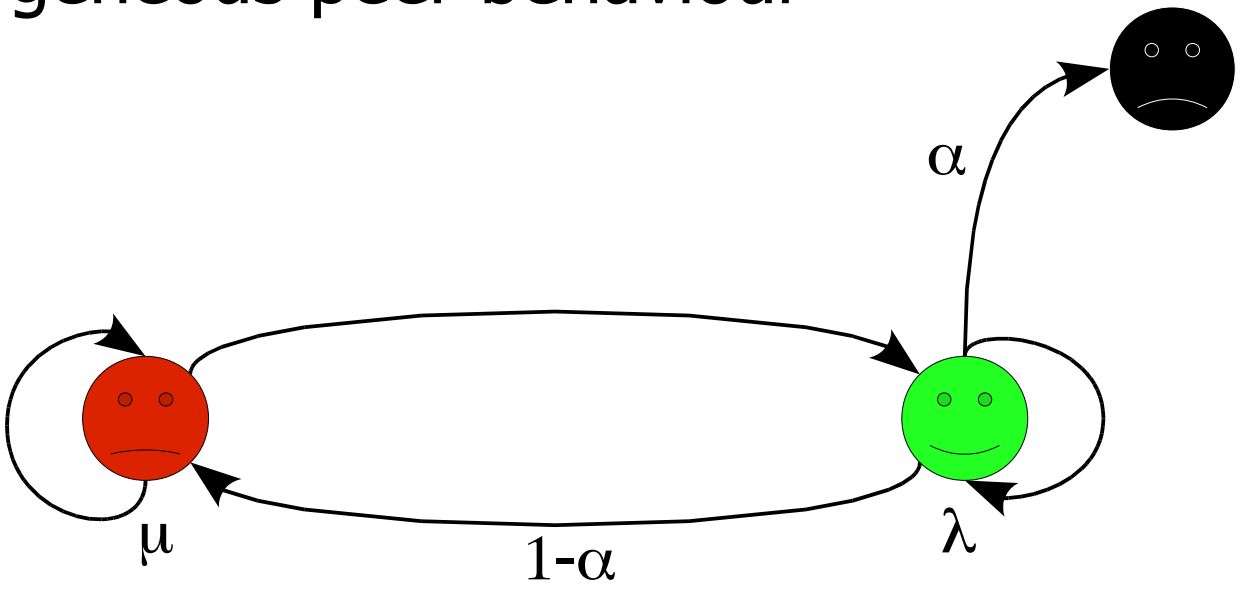
$$\gamma = failure\ rate\ ,\ f(t) = \gamma e^{-\gamma t}$$

$$Mean\ Time\ To\ Failure\ (MTTF) = \frac{1}{\gamma}$$

$$Proba\ (Survival\ after\ t) = e^{-\gamma t} = e^{\frac{-t}{MTTF}}$$

Stochastic Model

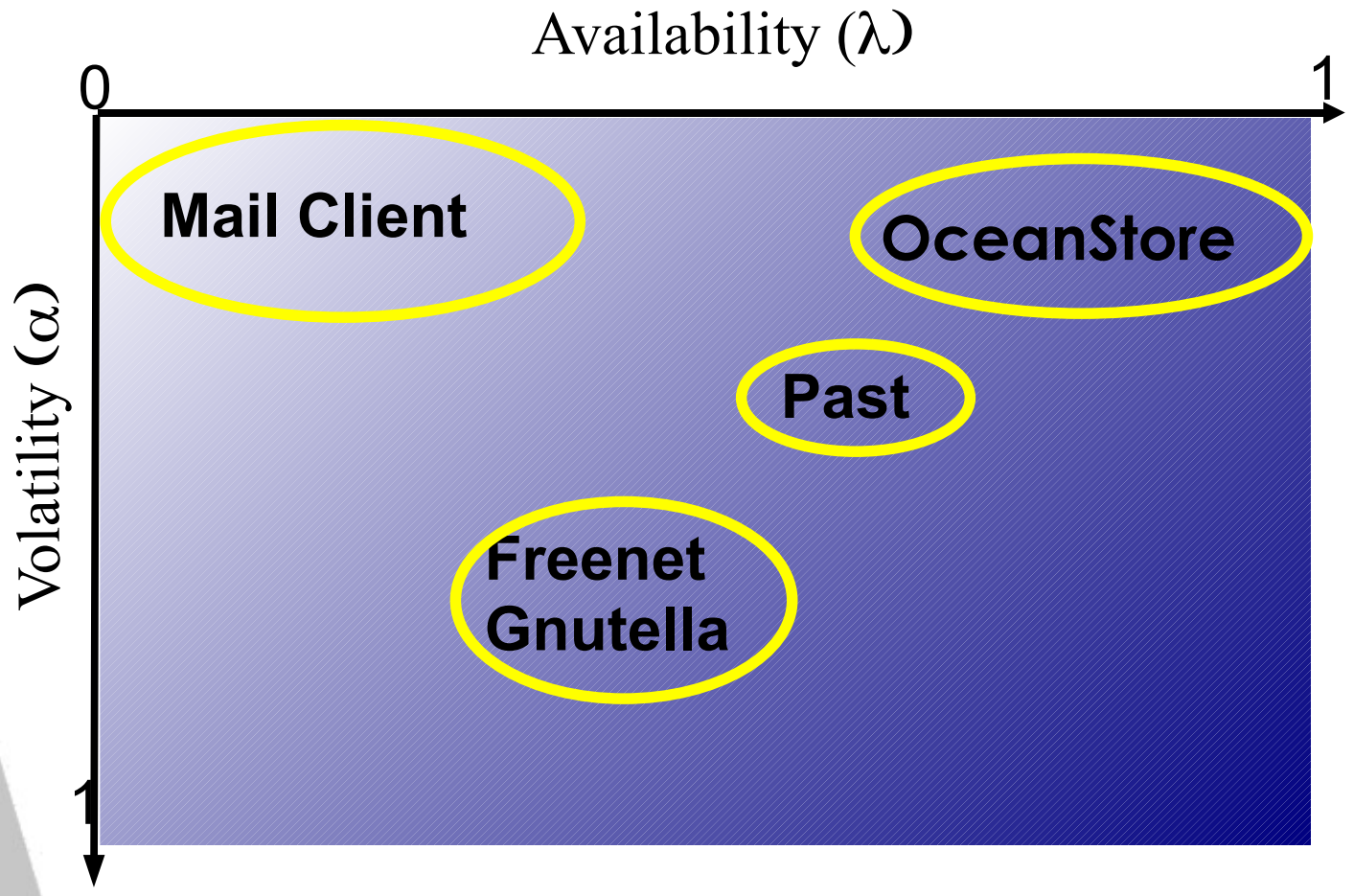
- Homogeneous peer behaviour



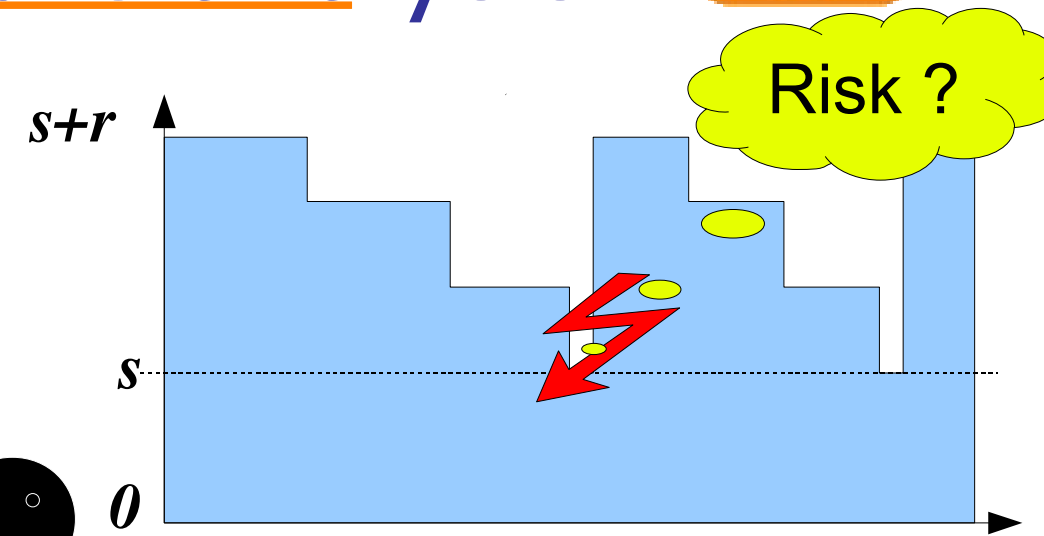
- Parameters:
  - α : Volatility
  - λ● : Availability

- Lifetime (MTTF) :
  - ●☞❍λ + μ = 1

System Characterization

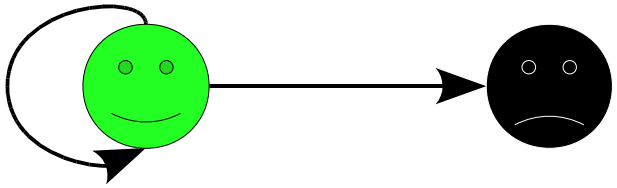


Reconstruction failure analysis



Hypothesis:

- storer states available,dead
- MTTF ***m***



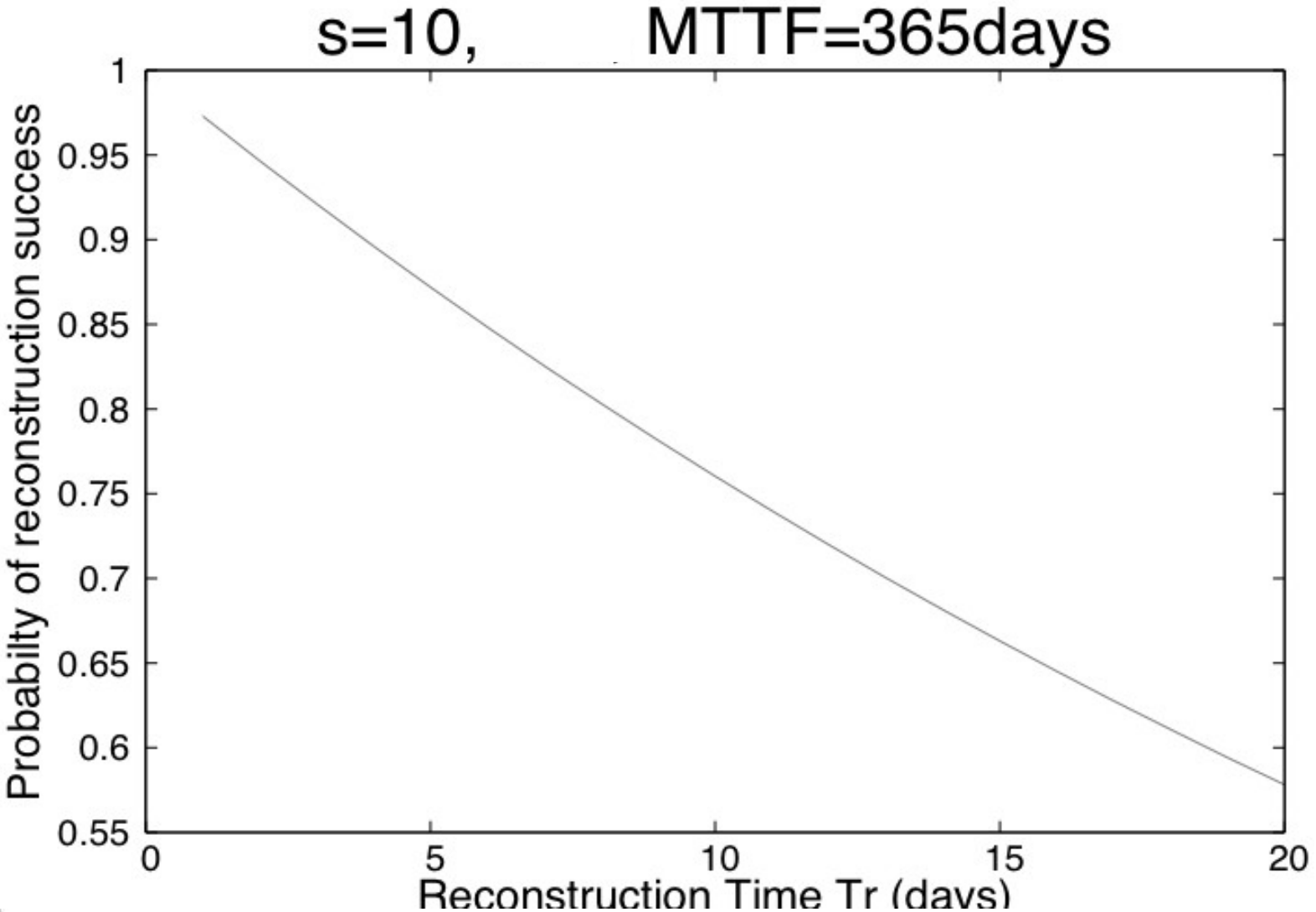
- ***Tr:*** needed time to reconstruct missing fragments,

$$P(X > Tr) = e^{-t/m}$$

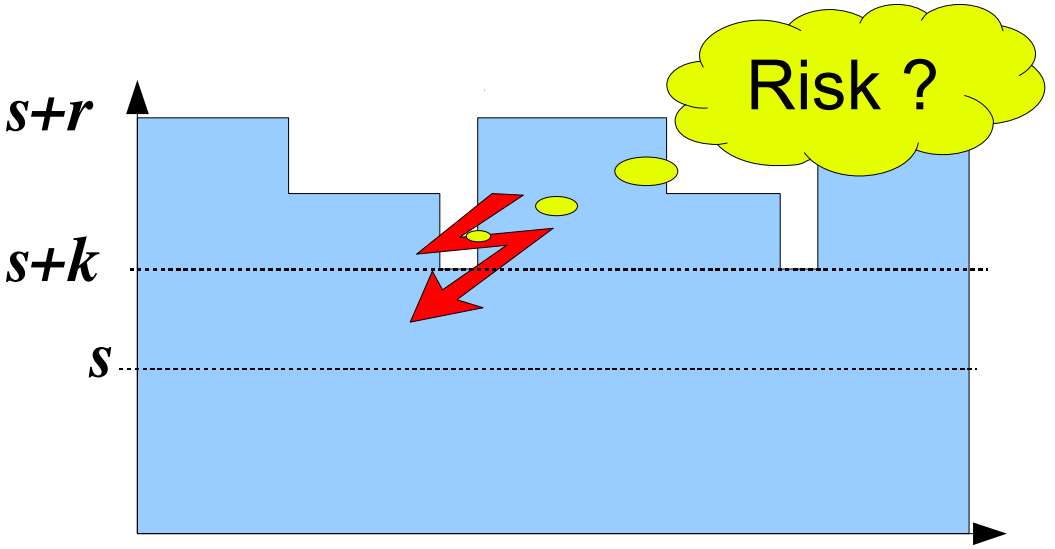
Probabilty of reconstruction success ?

Exemple: m = 1 year, Tr = 1 day, s = 10, k = 0, p = ?

Reconstruction failure analysis



Reconstruction failure analysis

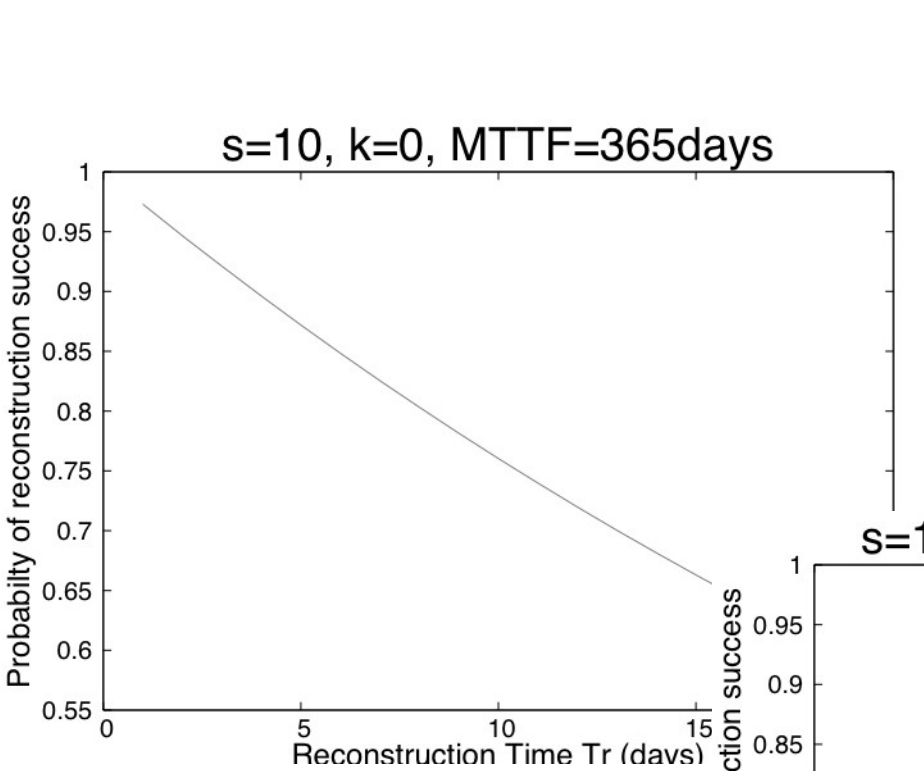


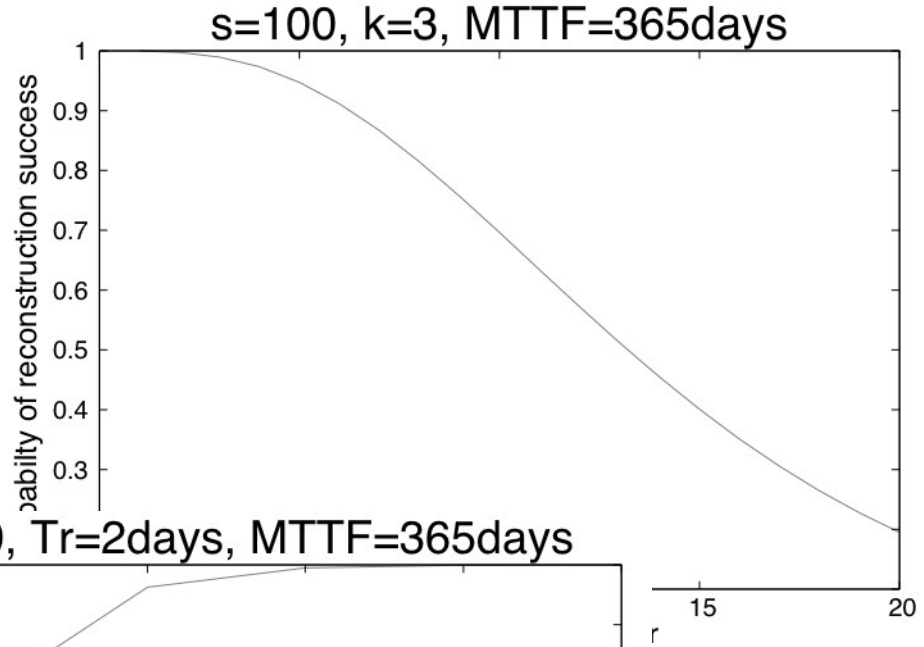
Why not reconstruct at ***s+k ?***

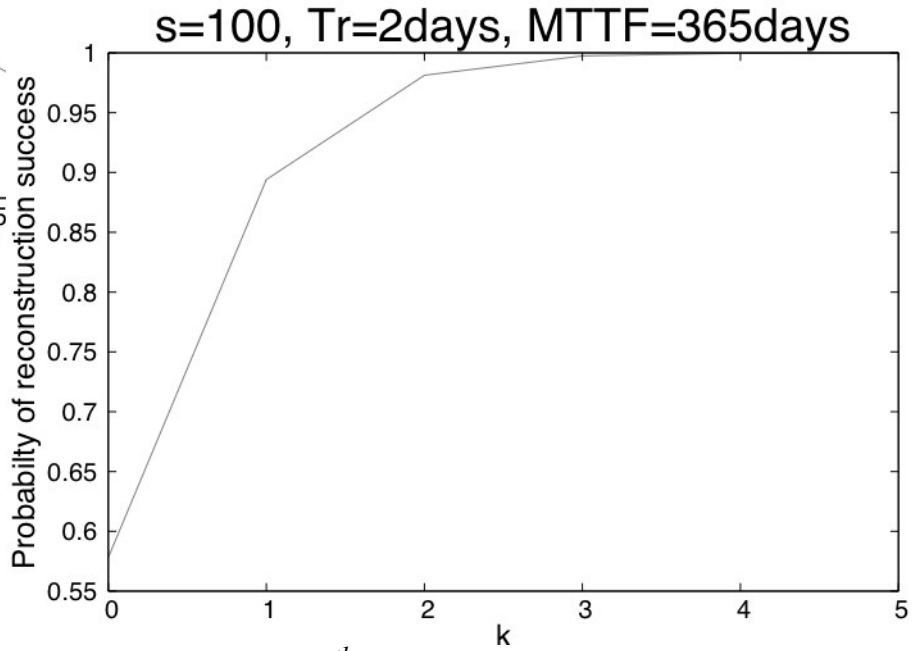
- Hypothesis:
  s+k remaining peers
- Probabilty of reconstruct success ?

$$\sum_{i=0}^{k} C_{s+i}^{s+k} \left(P(X>Tr)\right)^{s+i} \left(1-P(X>Tr)\right)^{k-i}$$

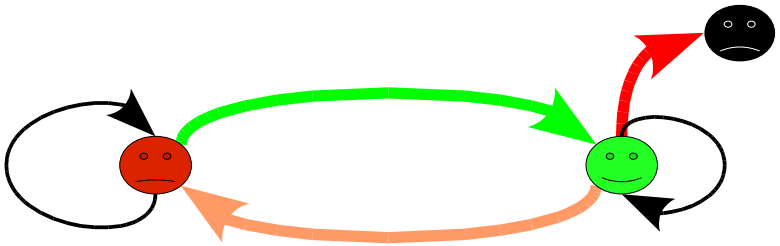
Reconstruction failure analysis

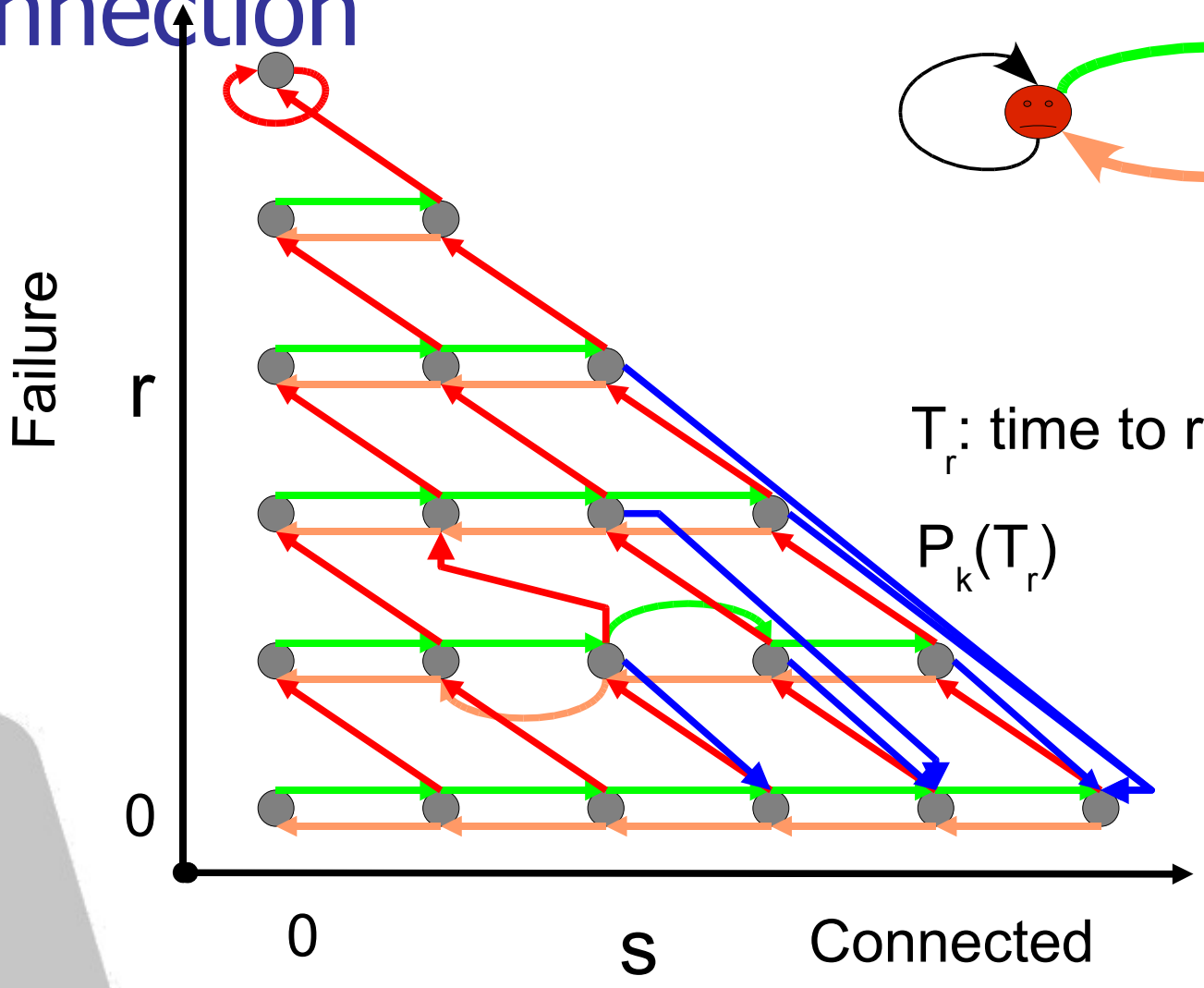






Data bloc Markov Process with disconnection





T_r: time to repair fragment

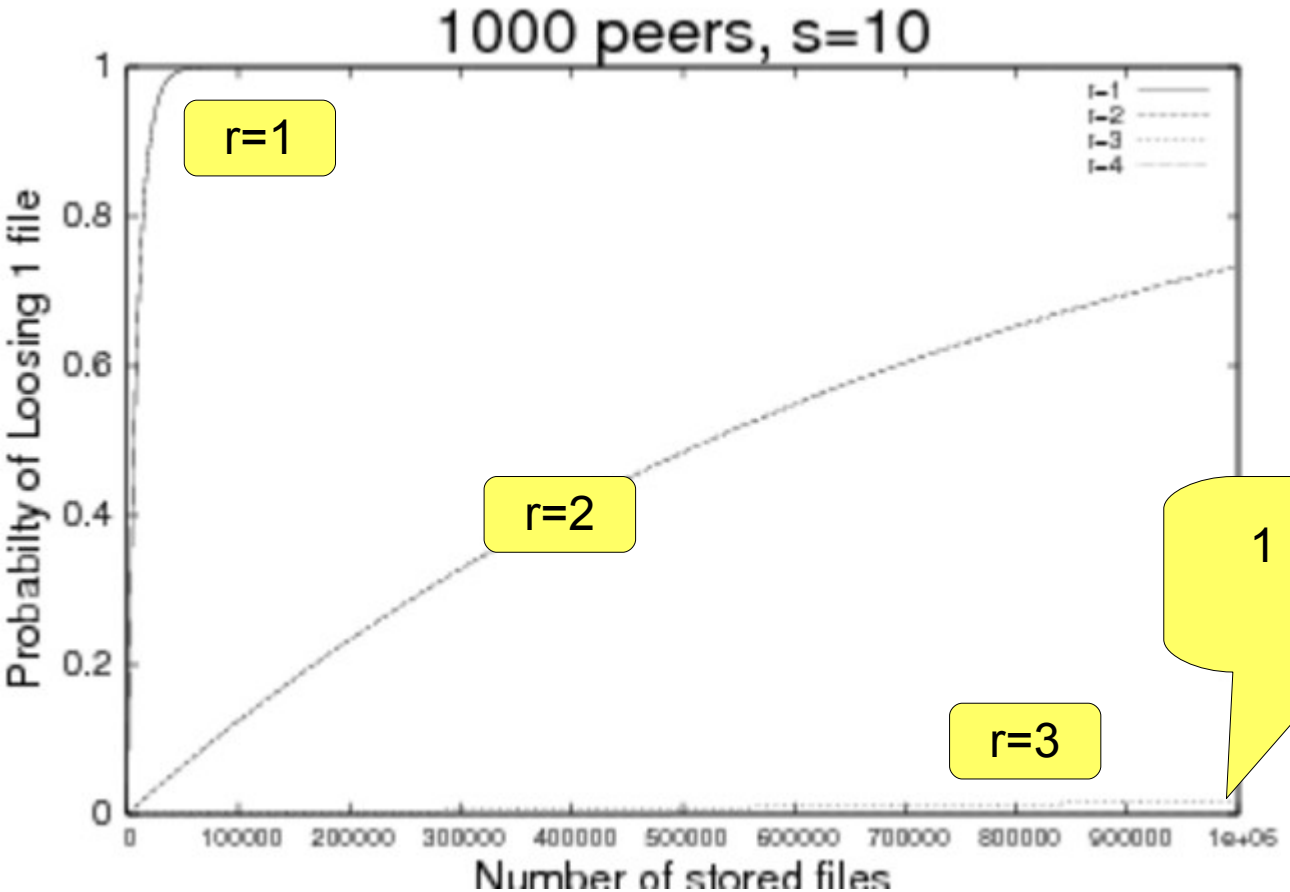
$P_k(T_r)$

Reed Solomon global system point of view

- Suppose r + 1 peers fails in the same time (DOS attack for instance)
- what is the probability of loosing one file ?
  - 10 peers, s=2, r=2, 1 file, p=?
  - UnFavorable cases : $C_{n-(r+1)}^{s-1}$
  - Possible cases : C_n^{s+r}

$$Probabilty\ of\ loosing\ 1\ stored\ file = 1 - \frac{C_{n-(r+1)}^{s-1}}{C_n^{s+r}}$$

Probabilty of loosing 1 file

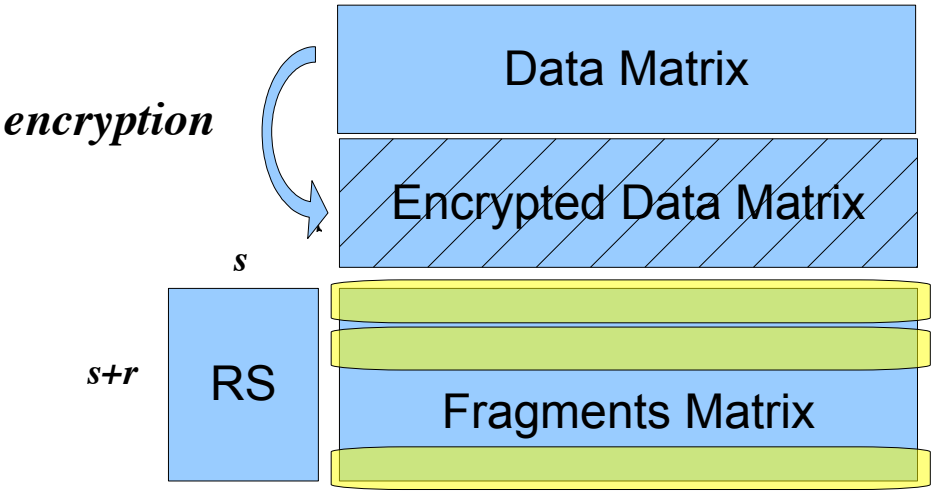


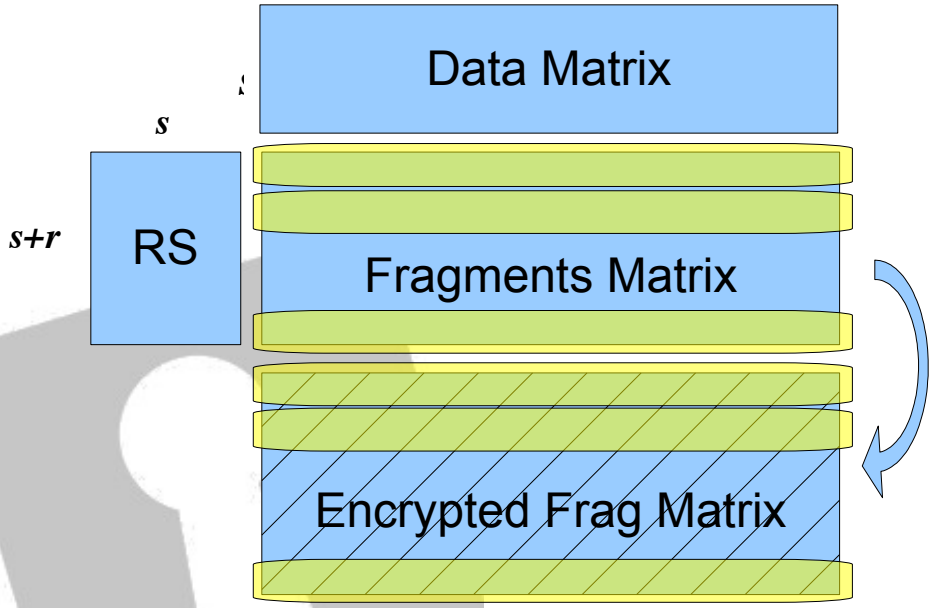
UbiStorage

Confidentiality: Keeping files secret

Making files secret

- For storage/backup, only client should read its spreaded files.
  - encrypt files ?
    - fragments maintenance can be done without knowing about encryption

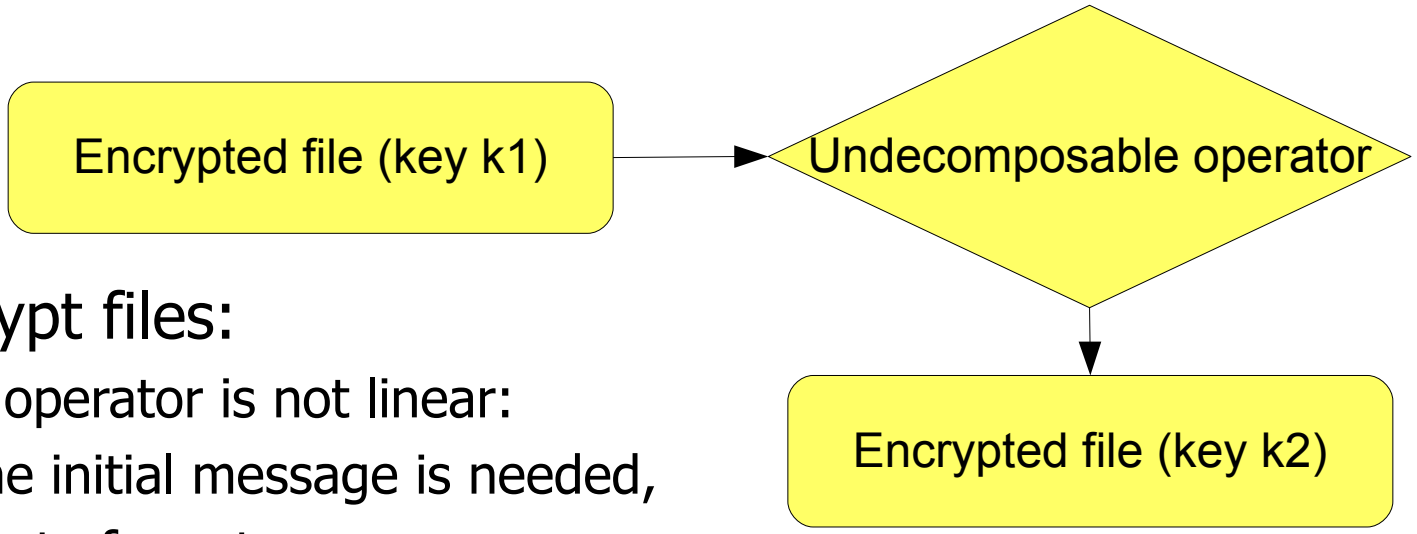




- encrypt fragments ?
  - how to rebuild missing fragments without knowing about encryption ?
  - does an asymmetric encryption can help ?

Re-encryption

- What about changing encryption key for spreaded files ?
- « Re-encryption protocol » transform one ciphertext to another without ever revealing the plaintext.



- encrypt files:
  - if operator is not linear:
  - the initial message is needed,
  - cost of a get...
- encrypt fragments:
  - just send the operator to storers

UbiStorage

Reed Solomon for storage perspectives

ReedSolomon: perspectives

- use hybrid RS-Replication to reduce redundancy maintenance costs and put delays
  - counter part is useful space cost
  - problem of analysis
- how to decide about peer storer death ?
- study of dynamic peers with proactive reconstruction
- ...

UbiStorage

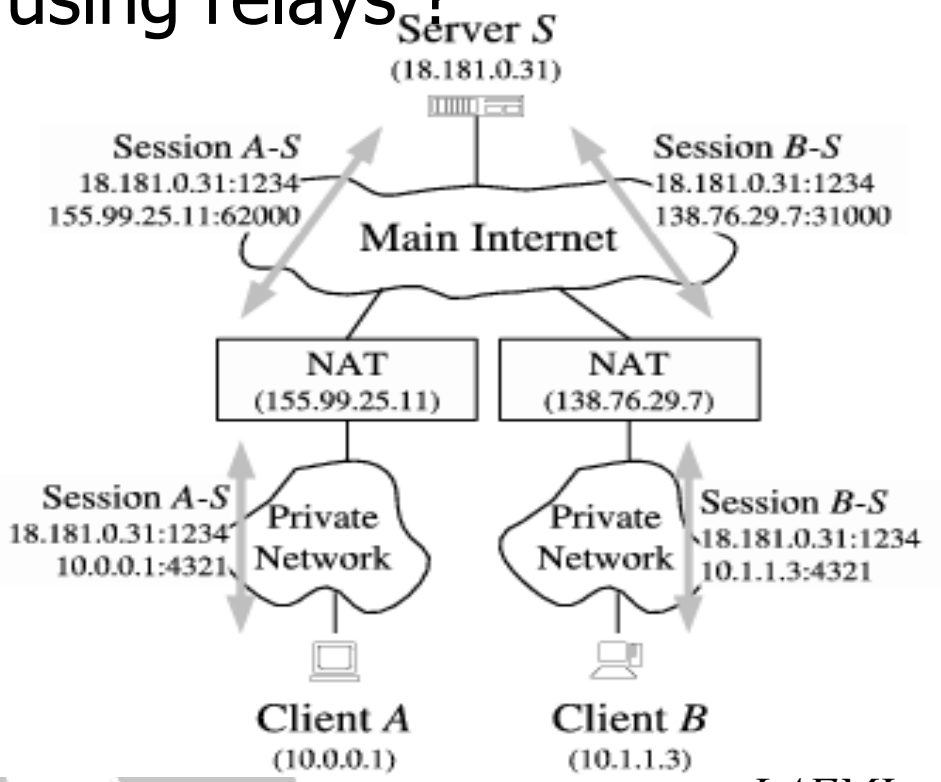
Connectivity

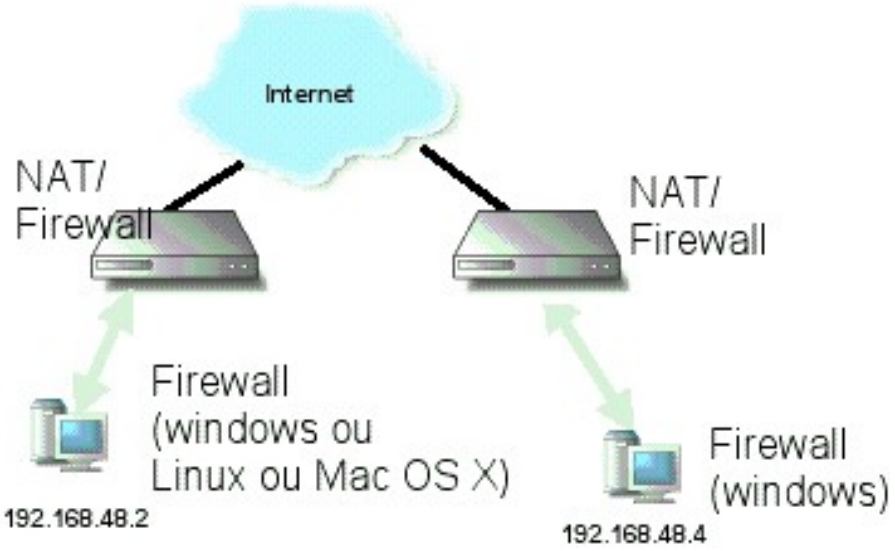
Connectivity between peers

(slides from F. Cappello JRES2005)
http://www.brynosaurus.com/pub/net/p2pnat/

- NAT and FireWall is a problem for direct communications

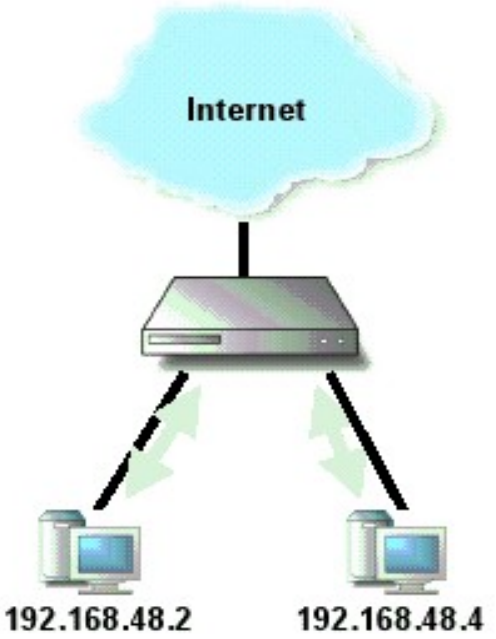
using relays ?





Passing thru NAT

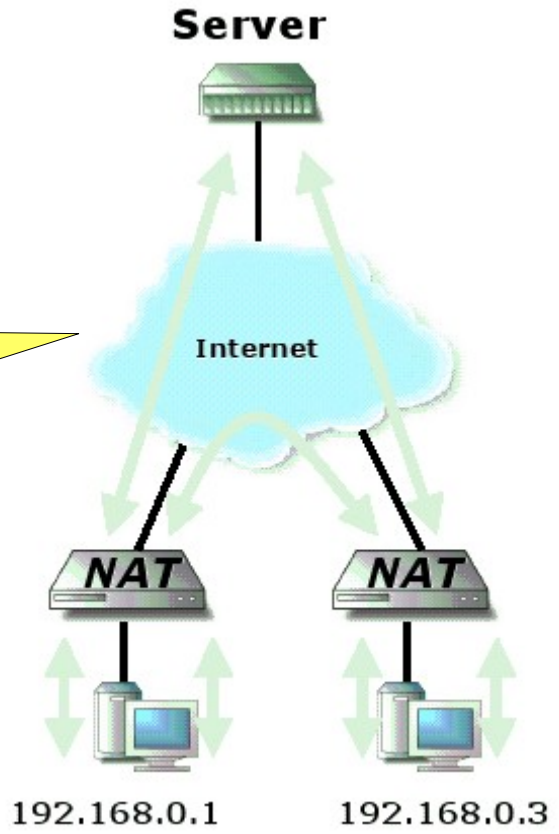
- Dynamic Routeur Configuration
  - UPNP (Universal Plug And Play)
    - router discovery
    - for each new connection:
      - add specific forwarding rule in the router
      - direct communication with external peer
    - at the end of the connection:
      - remove the added rule in the router
    - often disabled for security issues
  - Condor for firewall with agents



Passing thru NAT

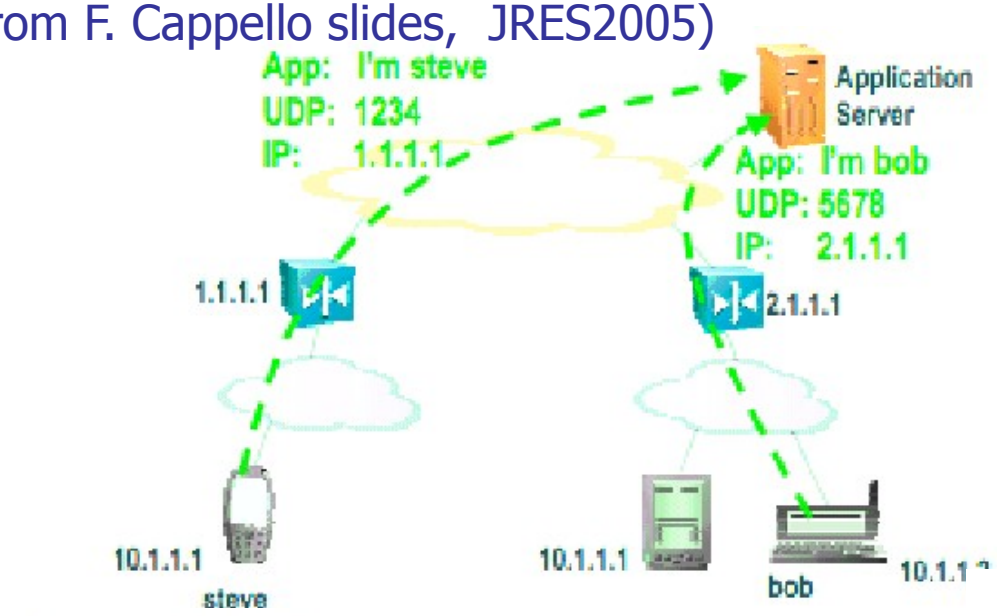
- Playing with NAT and UDP using a 'meeting' server allow to establish direct connection:

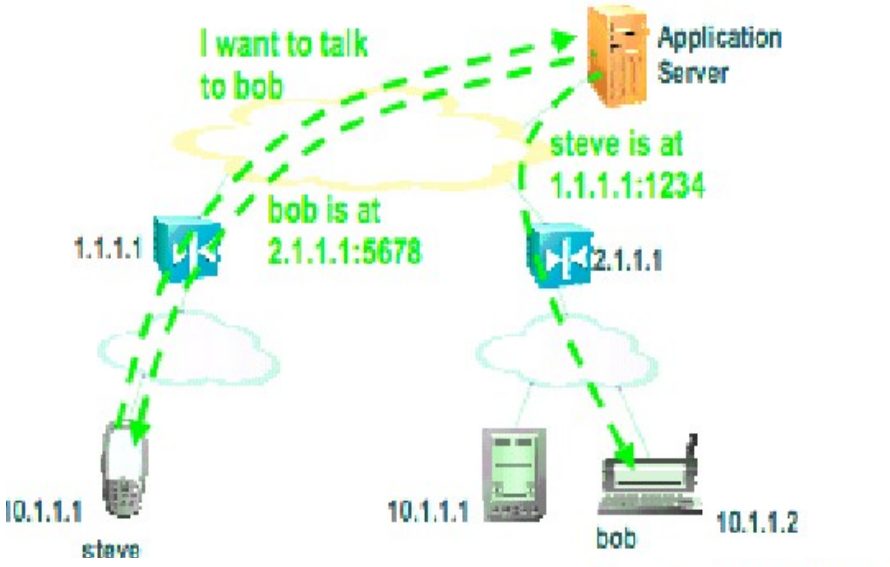
Hole Punching

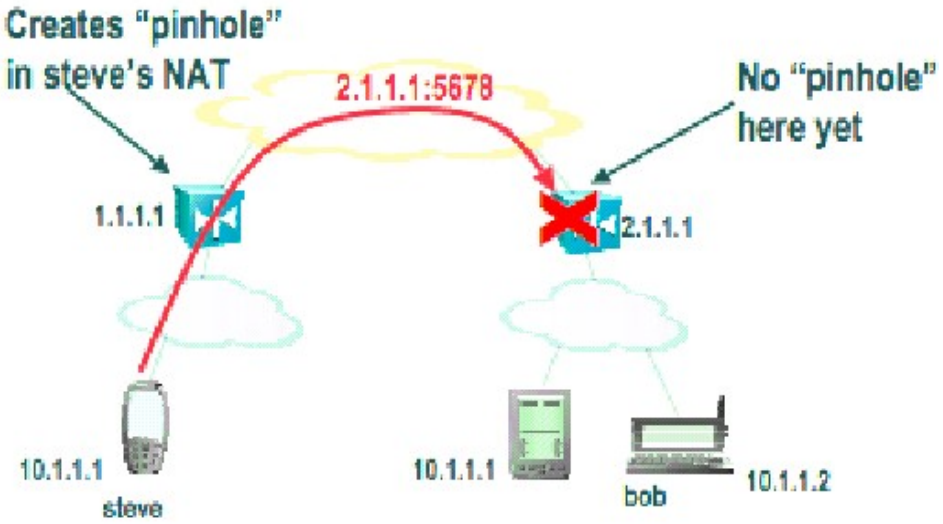


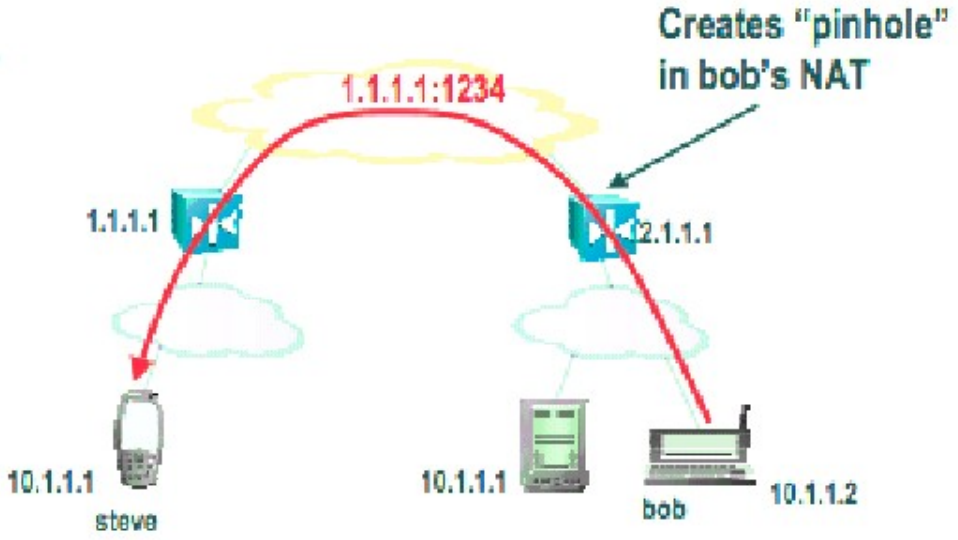
Hole Punching

(from F. Cappello slides, JRES2005)



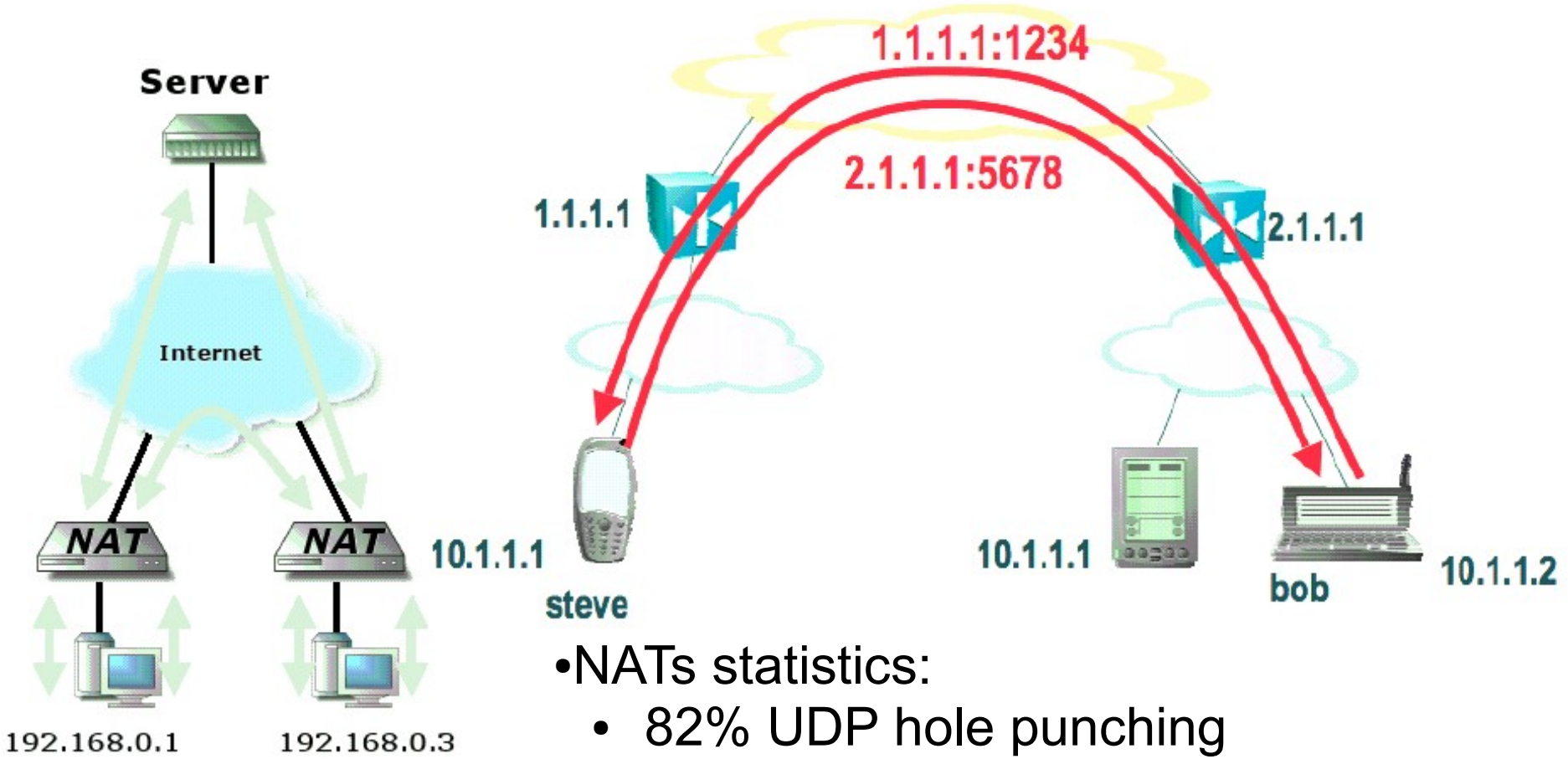






Hole Punching

(from F. Cappello slides, JRES2005)



- NATs statistics:
  - 82% UDP hole punching
  - 64% TCP hole punching
- (source http://www.brynosaurus.com/pub/net/p2pnat/, 2005)

Hole Punching

- RFC 3489 : http://www.ietf.org/rfc/rfc3489.txt
  - STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)
- STUNT : http://nutss.gforge.cis.cornell.edu/stunt.php
  - Simple Traversal of UDP Through NATs and TCP too
  - C API with Java Bindings
- JSTUN : http://jstun.javawi.de/
- ...

UbiStorage

Localisation

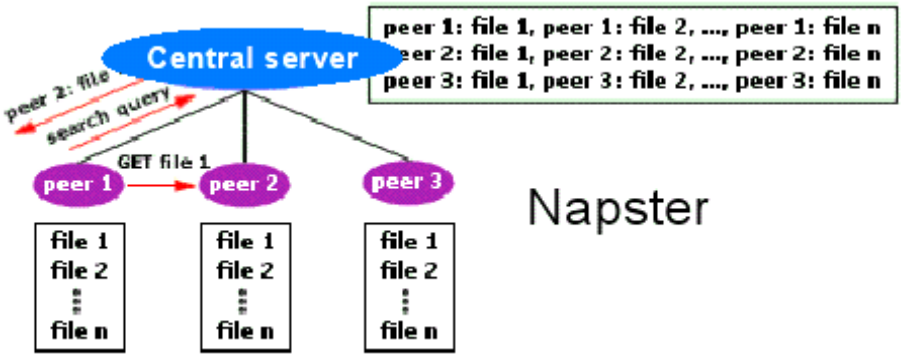
Information Localisation

- How to know information about stored files ?
- How to know where fragments are ?
- How to know which files are in critical state ?
- Expected reliability of this information ?

Information Localisation

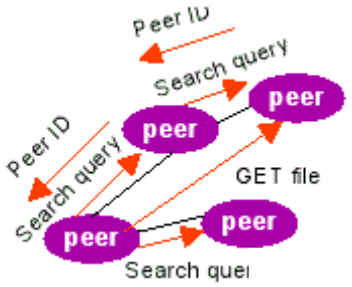
(from F. Cappello slides, JRES2005)

1st Generation : centralized index



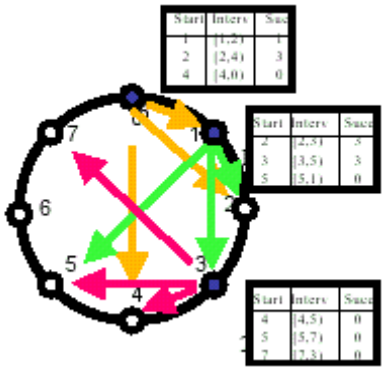
Napster

2nd Generation : innondation



Gnutella,

3th Generation : DHT Distributed Hash Table



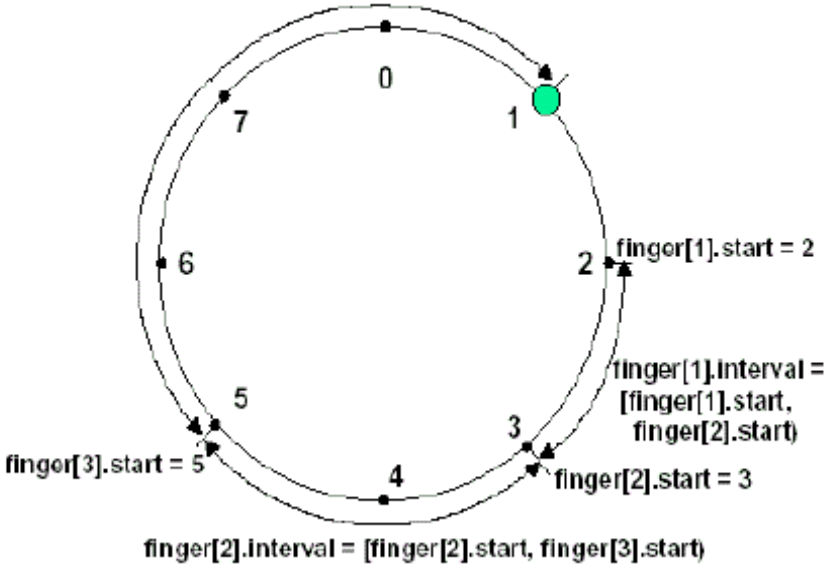
CAN, Chord, Pastry, etc.

DHT

- Distributed:
  - no server, no global index
  - Efficiency of primitive actions:
    insert, remove, search
  - each node is responsible of a set of keys (independant of the number of nodes)
- Hash Table:
  - Addressing associations <Key, Value>
- Space of key adressing is related to a specific structure which allow efficiency actions.
- Many work around DHT
  - routing in DHT, overlay maintenance, dynamicity properties...

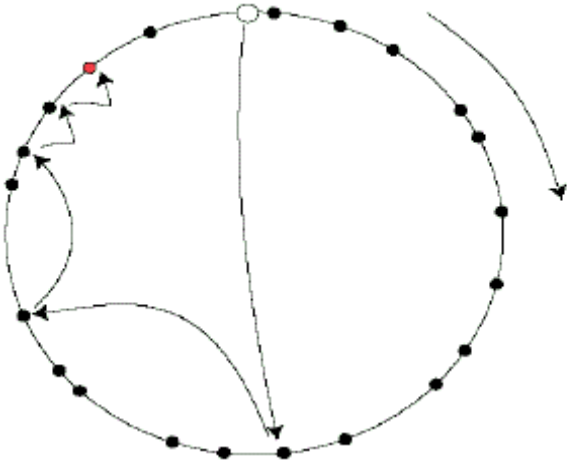
DHT Overlay

- Maintain the overlay structure of peer which defined the key location and routing scheme
- Example in Chord:
  - Localisation:
    - nodesId and keys coded on m bytes
    - key k is assigned to node with nodeId k (or next)
    - to find the value, find the node with Id closest to k value

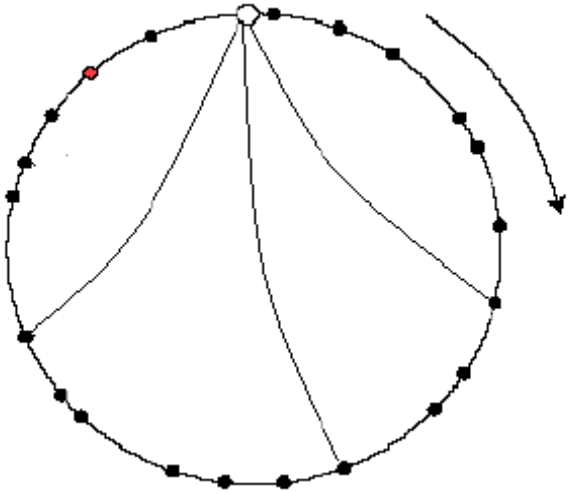


DHT Overlay

- Routing in Chord:
  - each node know m with ids
    $n+2^{i-1} \bmod 2^m$
  - to find k, jump to the closest node you know before k and so on



- Other technics:
  - random neighborhood (small world)



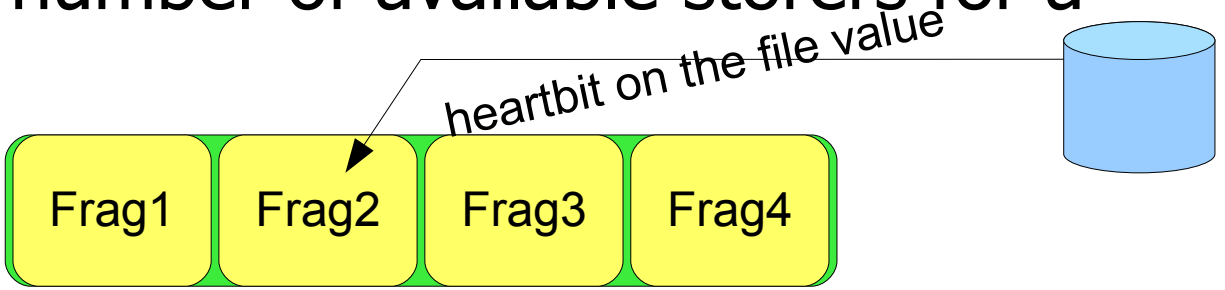
DHT Overlay Replication

- Data survability
  - copy information to the 'x' nodes which are are after k
- Many work around DHT performances
- About ReedSolomon:
  - can we use DHT to store information on files/fragments localisation, such that maintenance of Redundancy will be practicable ?

DHT for Reed Solomon Meta-Information

- Assume that value of <key,value> contains:
  - file and fragments information
- How ot know the number of available storers for a file ?

File associated value :
(fragment heartbit)



  - What will be the cost in the maintenance scheme ?
  - Combining ReedSolomon with replication with proactive reconstruction ?

biblio

- Performance analysis:
  - « Data Durability in Peer to Peer Storage Systems », Utard, Vernois, GP2PC03, 12-15 mai 2003 [ps]
  - « Performance Analysis of Peer-to-Peer Storage Systems », Alouf, Dandoush, Nain, Proc. of 20th International Teletraffic Congress (ITC 20), Ottawa, Canada, June 17-21, 2007. [pdf]
- ReedSolomon Implementation and usage in P2P storage:
  - « A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems », J.S. Plank, University of Tennessee 1996, [and others papers]
- Data distribution:
  - « Data distribution in a peer to peer storage system », Randriamaro, Soyez, Utard, Wlazinski. Journal of Grid Computing (JoGC), special issue on Global and Peer-to-Peer Computing, Springer, 2006 [ps]
  - « Dynamic distribution for data storage in a P2P network », Randriamaro, Soyez, Utard, Wlazinski, Grid and Pervasive Computing, mai 2007 [ps]
  - « Data distribution for failure correlation management in a Peer to Peer Storage », Randriamaro, Soyez, Utard, Wlazinski, International Symposium on Parallel and Distributed Computing, juillet 2005 [ps]
- ...
-

Known P2P storage/backup Solutions

CleverSafe http://www.cleversafe.org/

- OpenSource model (GPL), iSCSI interface

ActiveCircle http://www.active-circle.com/

- replication ?

Wuala http://wua.la/en/home.html

OceanStore

- http://oceanstore.cs.berkeley.edu/

...

Non-P2P solutions remain unexpensive...

example: https://www.mediamax.com/