sampling the join of streams

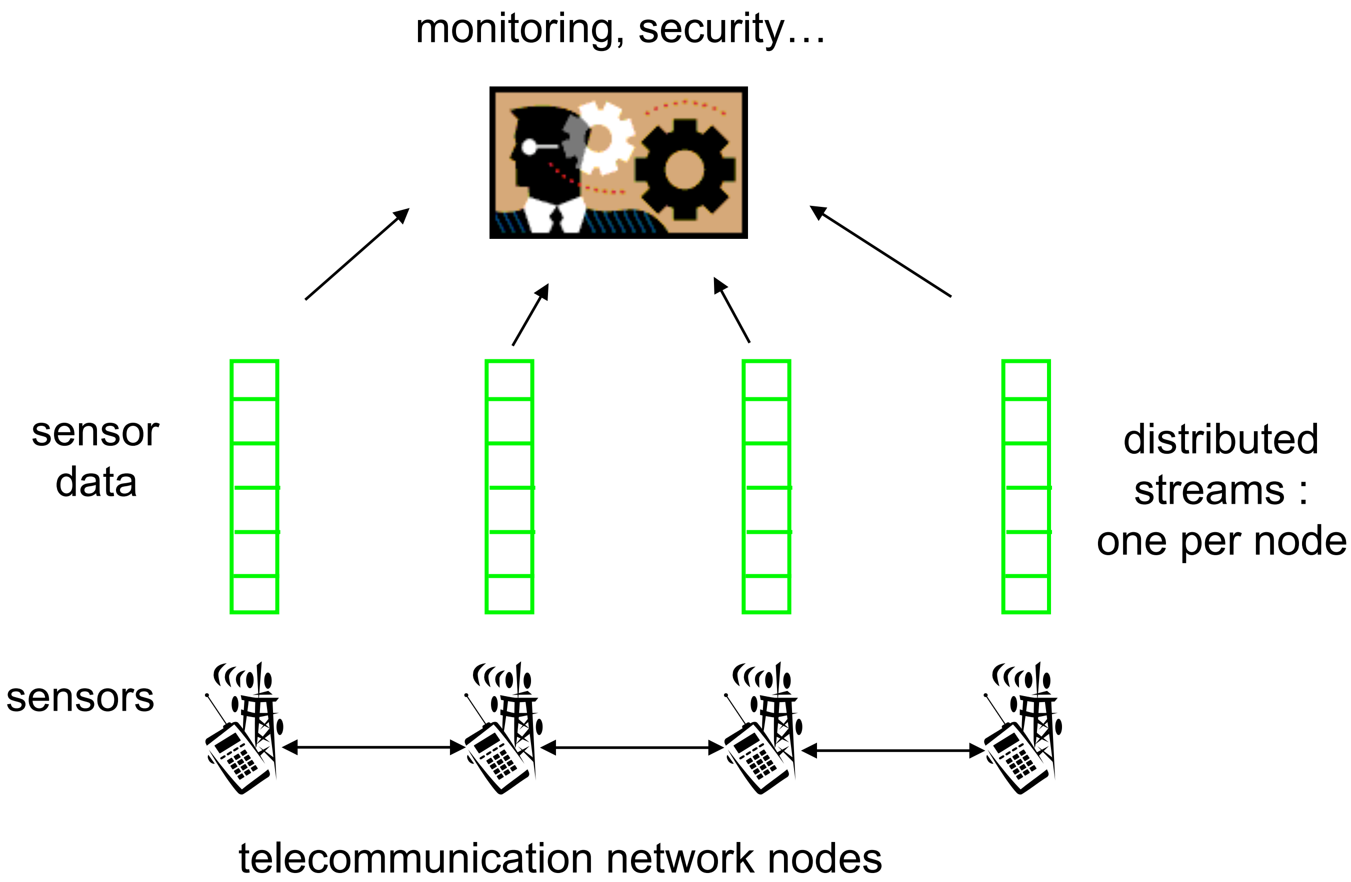
Raphaël Féraud : raphael.feraud@orange-ftgroup.com
Fabrice Clérot : fabrice.clerot@orange-ftgroup.com
Pascal Gouzien : pascal.gouzien@orange-ftgroup.com

outline

- problem
- four possible methods
- experiments & results
- conclusion

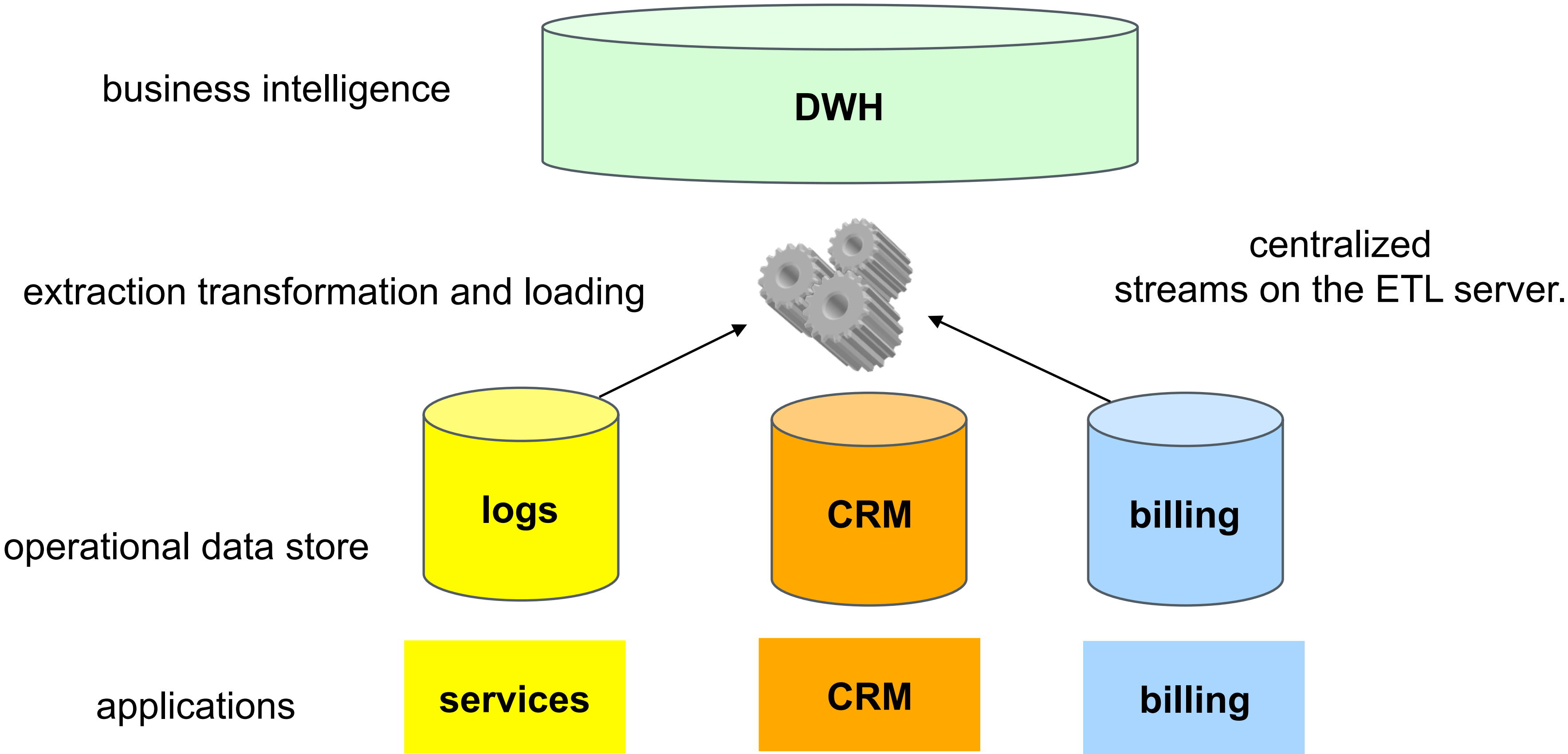
why join streams ?

- processing network sensor data



why join streams ?

- feed information system at lower cost : tables are processed as streams

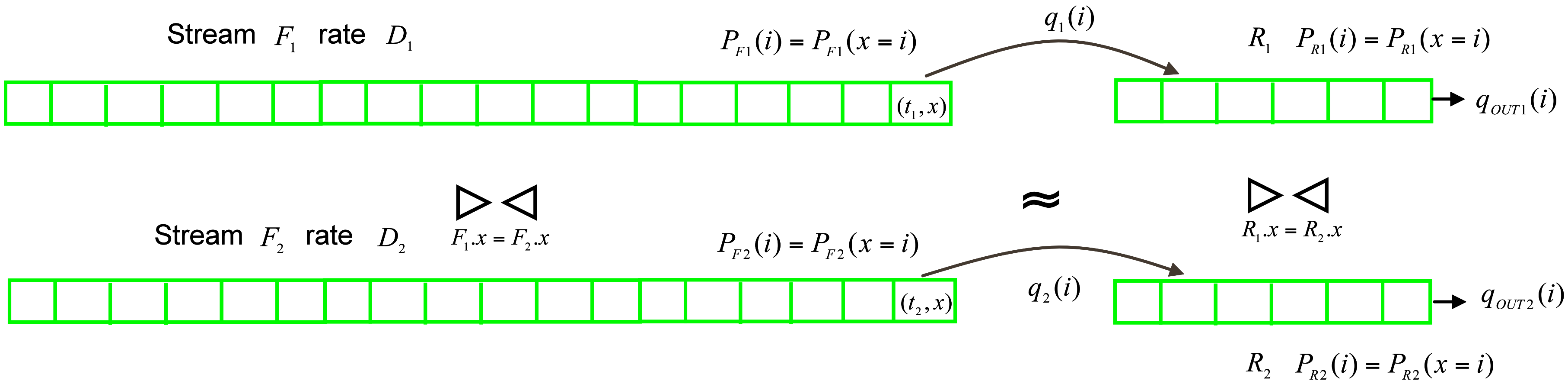


why sample the join of streams ?

- answer various queries on the join of two data streams F1 and F2 at any point t in time
- the join is $F_1^{<t} >< F_2^{<t}$
- the join operation is blocking
  - the join cannot be emitted as a flow without keeping F_1 and F_2 in memory
  - under finite memory constraint, the whole join cannot be produced.
- the join must be estimated from samples of F_1 and F_2

general framework

- the sample of the join consists in two reservoirs, which can be joined at any time.



- four probabilities per stream :
  - $P_F(i)$ probability of join key i in F
  - $P_R(i)$ probability of join key i in R
  - q(i) inclusion probability of join key i in R
  - $q_{out}(i)$ exclusion probability of join key i from R

outline

- problem
- four possible methods
  - Reservoir Sampling
  - Weighted Reservoir Sampling
  - Deterministic Reservoir Sampling
  - Active Reservoir Sampling
- experiments & results
- conclusion

reservoir sampling

- reservoir sampling algorithm (see Vitter 1985) evaluates
  - the inclusion probability as : q(t) = |R|/ (t+1)
  - the exclusion probability as : q_{out} = 1/|R|
  - both independent of the key and of the stream
- property :
  - when t tuples have been processed, the probability of each tuple to be in the reservoir is |R|/t
  - reservoir sampling allows to draw a uniform sample

outline

- problem
- four possible methods
  - Reservoir Sampling
  - Weighted Reservoir Sampling
  - Deterministic Reservoir Sampling
  - Active Reservoir Sampling
- experiments & results
- conclusion

motivation

- avoid wasting reservoir space by having keys in one reservoir not joining with keys in the other.
- in other words: find the sampling distributions P_{R1} and P_{R2} in the reservoirs that maximize the size of the join obtained by joining the two reservoirs.
- "careful with that axe, Eugene" … there are some constraints
  - respect the key distribution in the join !
  - the total size of the reservoirs is bounded

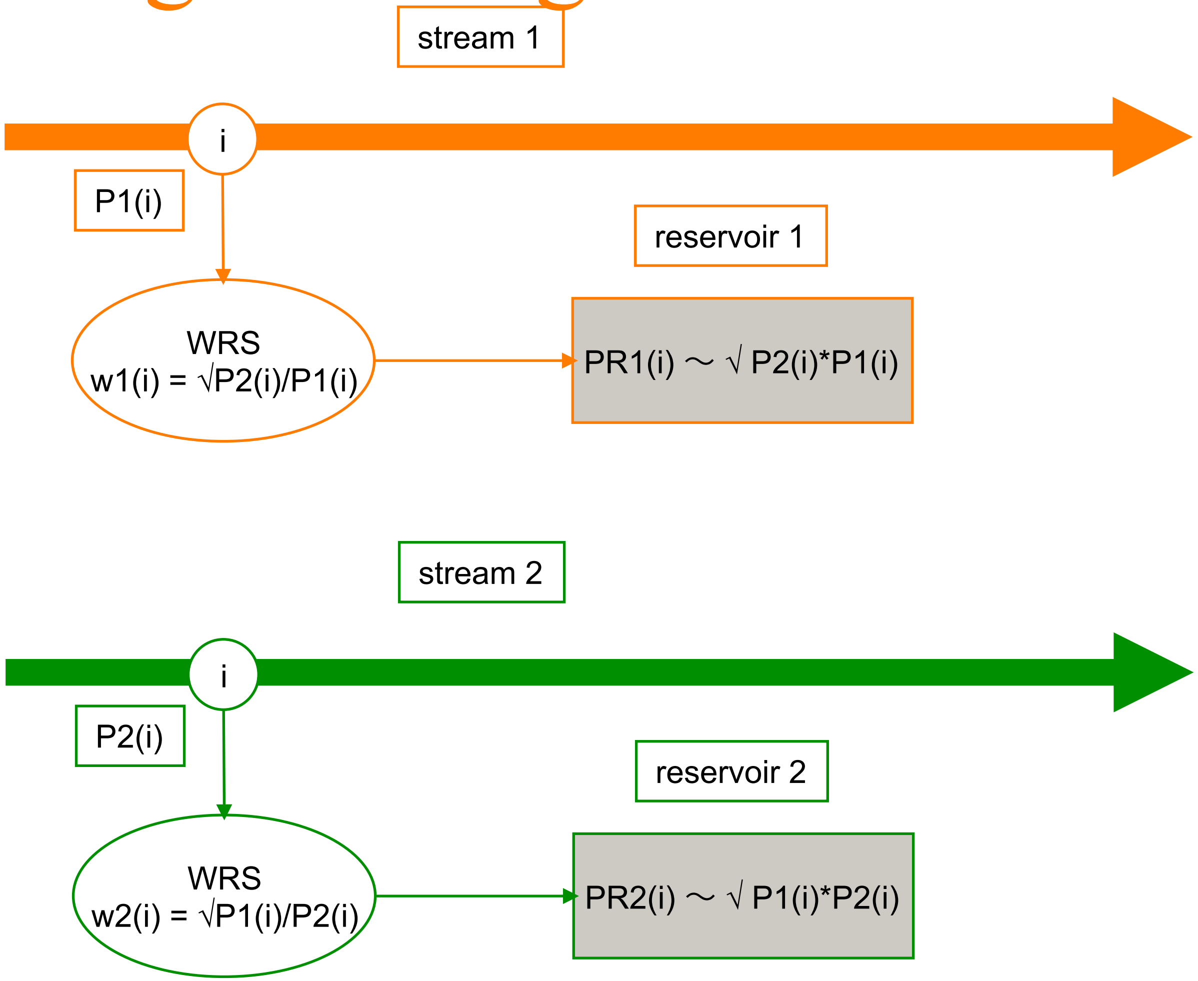
which value for $P_R(i)$?

- maximize the join size : $\Sigma_i |R_1|*P_{R1}(i) * |R_2|*P_{R2}(i)$
- under the constraints:
  - $P_{R1}(i)*P_{R2}(i)/\Sigma_j P_{R1}(j)*P_{R2}(j) = P_{F1><F2}(i)$
  - $R_1+R_2 = R$
- Optimal solution
  - **$R_1 = R_2 = R/2$**
  - **$P_{R1}(j) = P_{R2}(j) = [C^* P_{F1><F2}(i)]^{1/2} \sim [P_{F1><F2}(i)]^{1/2}$**
  - Where $C= \Sigma_j P_{R1}(j)*P_{R2}(j)$

how to reach the right value for $P_R(i)$?

- Weighted Reservoir Sampling (Kolonko 2004, Efraimidis 2005) allows to set $P_R(i)$ to a desired value :
  - read key i
  - draw v according to an exponential distribution with parameter $\lambda = P_R(i)$
  - if v < R[n].v then
    - delete R[n] // n is the size of R
    - insert (i,v) in R ordered by ascending v

setting the weights



bounding the join size

- using Weighted Reservoir Sampling, we have :
  - $mean(J(i)) = R_1 * P_{R1}(i) * R_2 * P_{R2}(i) = R/2)^2 * (P_1(i) * P_2(i)) / (\Sigma_j P_1(j) * P_2(j)^{1/2})^2$
  - $mean(J_{WRS}) = (R/2)^2 * (\Sigma_j P_1(j) * P_2(j)) / (\Sigma_j (P_1(j) * P_2(j))^{1/2})^2$
- using Reservoir Sampling, we have :
  - $mean(J_{RS}) = (R/2)^2 * (\Sigma_j P_1(j) * P_2(j))$ (with two reservoirs of equal size R/2)
- remark:
  - $(\Sigma_j P_1(j) * P_2(j)) / (\Sigma_j (P_1(j) * P_2(j))^{1/2})^2 \leqq 1$
  - under the constraints $\Sigma_j P_k(j) = 1$, $\Sigma_{j=1} (P_1(j) * P_2(j))^{1/2} \leqq 1$
- therefore **$mean(J_{RS}) \leqq mean(J_{WRS}) \leqq (R/2)^2$**

RS, WRS

- advantages
  - no communication between reservoirs
  - reservoirs a priori bounded
  - "anytime" sampling
- drawbacks
  - no fine tuning of the sample due to the independent sampling processes

outline

- problem
- four possible methods
  - Reservoir Sampling
  - Weighted Reservoir Sampling
  - Deterministic Reservoir Sampling
  - Active Reservoir Sampling
- experiments & results
- conclusion

Deterministic Reservoir Sampling

- Deterministic Reservoir Sampling is a three steps algorithm :
  - at the first step a sampling design based on the join key is built.
  - at the second step the samples R1 and R2 are collected.
  - at the third step the obtained samples R1 and R2 are optimised.

- the size of the join J is fixed.
- the size of the memory needed |R|=|R1|+|R2| depends on the distribution of join key
  - unknown in advance, except that |R|<|J|.

Deterministic Reservoir Sampling first step

- frequencies of the join key are supposed to be known (or well estimated) for both streams :

$$P(k) = \frac{P_1^k . P_2^k}{\sum_j^N P_1^j . P_2^j}$$

- the sampling design consists of the draw of |J| keys according to the distribution of the join key in the join :
  - J=0
  - for i=1 to |J| do
    - draw a key k from P(k)
    - J=J+{k}

Deterministic Reservoir Sampling second step

- the second step consists of the collect of the |J| desired keys :
- R1=0
- R2=0
- while |R1|.|R2| < |J|
  - if a key k arrives from F1
    - if |R1(k)| < |J(k)| and |R1(k)|.max(1,|R2(k)|) < |J(k)| then R1=R1 U {k}
  - if a key k arrives from F2
    - if |R2(k)| < |J(k)| and |R2(k)|.max(1,|R1(k)|) < |J(k)| then R2=R2 U {k}
- warning: the current state of each reservoir must be known to the other at every arrival

Deterministic Reservoir Sampling third step

- the purpose of the last step is to avoid rounding error between the obtained samples R1 and R2 and the sampling design J.
- it is done only when a query is requested on the join.
- for all join key i in $|R_1| \bowtie |R_2|$

$$E_1 = \left(|R_2^i|.\left(|R_1^i| - 1\right) - |J^i|\right)^2$$

$$E_2 = \left(|R_1^i|.\left(|R_2^i| - 1\right) - |J^i|\right)^2$$

$$E = \left(|R_1^i|.|R_2^i| - |J^i|\right)^2$$

  - while $\left(E_1 < E \text{ AND } E_1 < E_2 \text{ AND} |R_1^i| > 1\right) \text{OR} \left(E_2 < E \text{ AND } E_2 < E_1 \text{ AND} |R_2^i| > 1\right)$
    - $\text{IF} \left(E_1 < E_2\right) \text{THEN } |R_1^i| = |R_1^i| - 1$

      $\text{IF} \left(E_2 < E_1\right) \text{THEN } |R_2^i| = |R_2^i| - 1$
    - evaluate E, E_1 and E_2

Deterministic Reservoir Sampling

- assuming that we succeed to collect the samples needed R1 and R2, a sample drawn from the join key distribution is obtained
- therefore, DRS algorithm leads to the smallest variance of the key distribution in the join

- drawbacks
  - intense communication between reservoirs
  - no a priori "tight" bound on the sizes of the reservoirs
  - no bound on the time needed to fill up the reservoirs

deterministic sampling: another possible approach

- in the above approach, it is necessary for the two samplers to communicate intensively
- here is another approach:
  - draw a set of |J| keys from the key distribution in the join: $N_{12}(i)$
    - size of the sample of the join known apriori
  - define the optimal sampling design $N_a(i)$ on each stream separately so as to obtain the sample of the join
    - minimize $\Sigma_i \, (N_{12}(i) - N_1(i)*N_2(i))^2/N_{12}(i)$ with $\Sigma_i \, N_a(i) \leqq R_a$ (a=1,2)
    - the solution of the optimisation problem gives the size of each reservoir and the keys to collect in each reservoir
  - collect the keys independently on each stream until both reservoirs are complete

"light" deterministic sampling

- advantage:
  - no communication between reservoirs
  - reservoir sizes known a priori

- drawback:
  - no bound for the time necessary to complete the reservoirs, ie no bound for the time necessary to get the sample of the join

outline

- problem
- four possible methods
  - Reservoir Sampling
  - Weighted Reservoir Sampling
  - Deterministic Reservoir Sampling
  - Active Reservoir Sampling
- experiments & results
- conclusion

Active Reservoir Sampling

- we want a sample where the distribution of the join key is as close as possible of the true one.
- the idea is to minimize the χ^2 between $P_{R1><R2}$ and $P_{F1><F2}$

$$Khi^2(P_{F1F2}, P_{R1R2}) = |R1 \triangleright\triangleleft R2| \sum_i \frac{(P_{R1R2}(i) - P_{F1F2}(i))^2}{P_{F1F2}(i)}$$

- by controlling inclusion probabilities $q_1(i)$ and $q_2(i)$
- exclusion probabilities are uniform

Active Reservoir Sampling

$$\frac{\partial Khi^2(P_{F1F2}, P_{R1R23})}{\partial q_1(k)} = \sum_i \frac{|R_1 \rhd\lhd R_2|}{P_{F1F2}(i)} \frac{\partial K(i)}{\partial q_1(k)}$$

$$K(i) = (P_{R1R2}(i) - P_{F1F2}(i))^2 = \left(\frac{P_{R1}(i) P_{R2}(i)}{\sum_l P_{R1}(l) P_{R2}(l)} - P_{F1F2}(i) \right)^2$$

$$\frac{\partial K(i)}{\partial q_1(k)} = \sum_j \frac{\partial K(i)}{\partial P_{R1}(j)} \frac{\partial P_{R1}(j)}{\partial q_1(k)}$$

- if we develop the previous equation we obtain two terms.

Active Reservoir Sampling

- the first term is obtained by direct derivation :

$$\frac{\partial K(i)}{\partial P_{R1}(j)} = -2(1-\lambda_{ij})\left(\frac{P_{R1}(i)P_{R2}(i)}{\sum_l P_{R1}(l)P_{R2}(l)} - P_{F1F2}(i)\right)\frac{P_{R1}(i)P_{R2}(i)P_{R2}(j)}{\left(\sum_l P_{R1}(l)P_{R2}(l)\right)^2}$$

$$+\,2\lambda_{ij}\left(\frac{P_{R1}(i)P_{R2}(i)}{\sum_l P_{R1}(l)P_{R2}(l)} - P_{F1F2}(i)\right)\frac{P_{R2}(i)\sum_l P_{R1}(l)P_{R2}(l) - P_{R1}(i)P_{R2}(i)^2}{\left(\sum_l P_{R1}(l)P_{R2}(l)\right)^2}$$

- Where λ_{ij} = 1 if i=j and 0 else

Active Reservoir Sampling

- Second term is obtained from fluid approximation (balance equation between inputs and outputs) :

$$\frac{\partial P_{R1}(j)}{\partial q_1(k)} = -(1-\lambda_{jk})\frac{D_1}{|R_1|}P_{F1}(k)\frac{|j|}{|R_1|} + \lambda_{jk}\frac{D_1}{|R_1|}\left(P_{F1}(k) - P_{F1}(k)\frac{|j|}{|R_1|}\right)$$

- Where λ_{ij} = 1 if i=j and 0 else

ARS

- advantages
  - extremely accurate control of the quality of the key distribution in the sample of the join
  - "anytime" sample
- drawbacks
  - computing intensive
  - communication intensive

outline

- problem
- four possible methods
- experiments & results
- conclusion

quality of the sampling

- the quality of the sampling is measured by :
  - the variance of the key distribution in the sample of the join
  - the size of the obtained sample of the join
  - the memory resources
- very important reminder:
  - the confidence interval on the result of a query on the sample of the join is a function of <u>both</u> variance and join size

- the following results are obtained using 100 draws for each value.

synthetic datasets

first toy problem

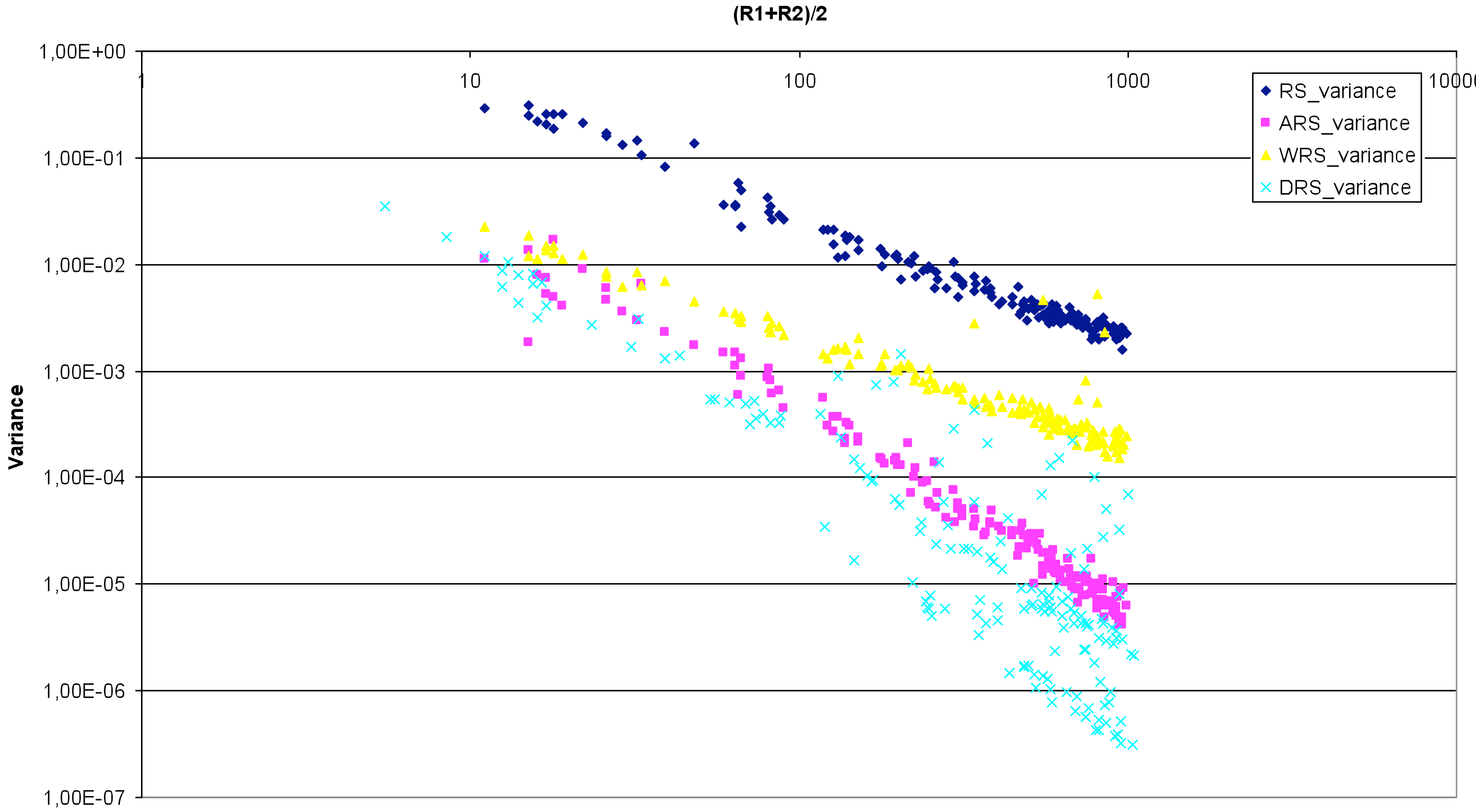
- two streams containing three join keys.
- each value of the join key have a different probabilities for each stream and for join stream :

key	Frequency table F1	
	Count	percentage
1	5050	5.00000
2	1010	1.00000
3	94940	94.00000

Key	Frequency table F2	
	Count	percentage
1	2020	2.00000
2	90900	90.00000
3	80800	8.00000

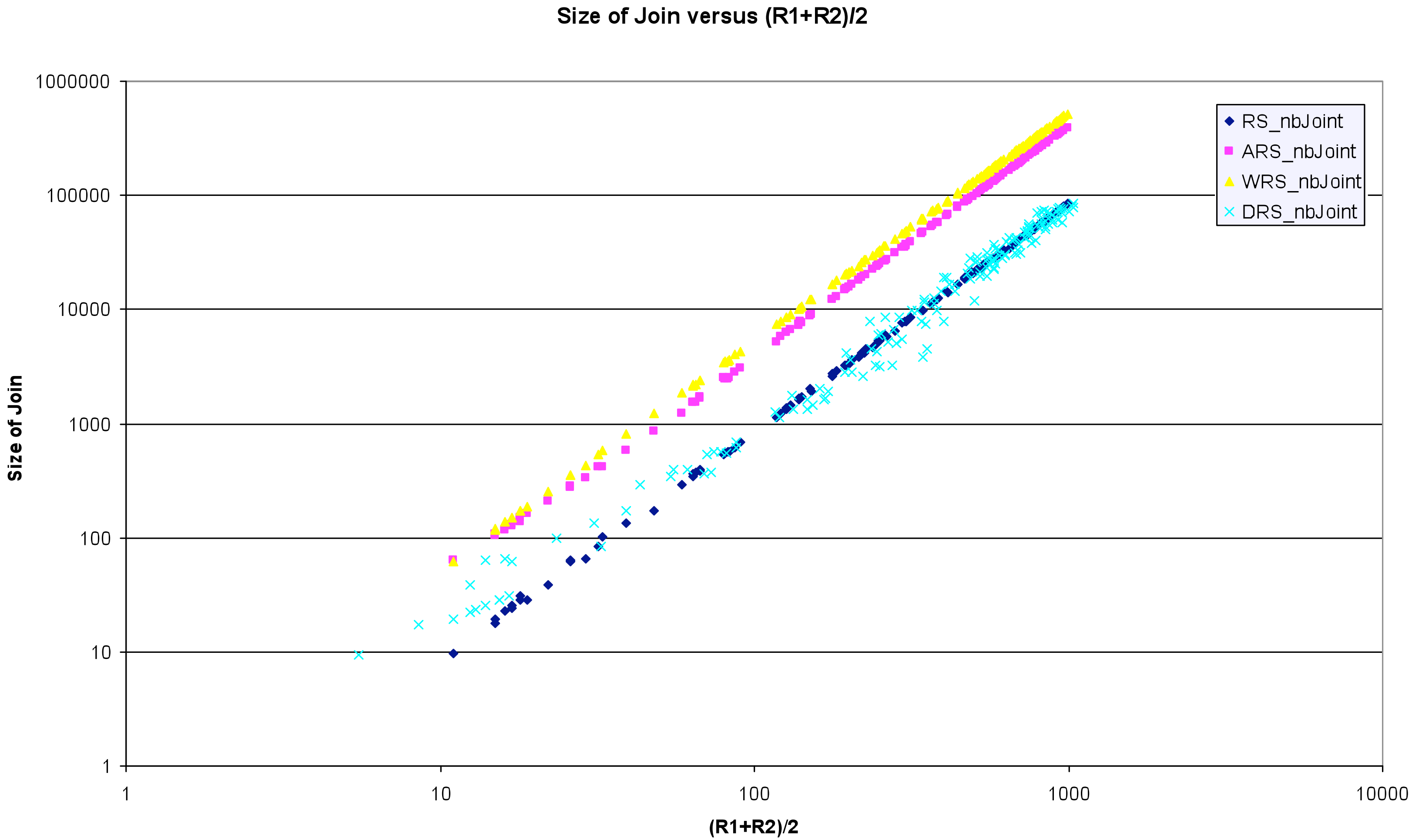
Key	Frequency table F1 >< F2	
	Count	percentage
1	10201000	1.17382
2	91809000	10.56332
3	7671152000	88.26286

variance



(R1+R2)/2
1,00E+00
1,00E-01
1,00E-02
1,00E-03
1,00E-04
1,00E-05
1,00E-06
1,00E-07
1
10
100
1000
10000
Variance
RS_variance
ARS_variance
WRS_variance
DRS_variance

join size


Size of Join versus (R1+R2)/2
1000000
100000
10000
1000
100
10
1
Size of Join
1
10
100
1000
10000
(R1+R2)/2
RS_nbJoint
ARS_nbJoint
WRS_nbJoint
DRS_nbJoint

first toy problem : conclusion

- Reservoir Sampling estimator is not robust : large variance and small size of join.
- Deterministic Reservoir Sampling estimator is robust : smaller variance than other estimators.
- Weighted Reservoir Sampling estimator leads to a large size of join, but with a variance higher than Deterministic Reservoir Sampling.
- Active Reservoir Sampling outperforms other algorithms : low variance, and large size of join for limited ressources but potentially very computing intensive: one update is $O(C^3)$ with C the number of keys ...

second toy problem

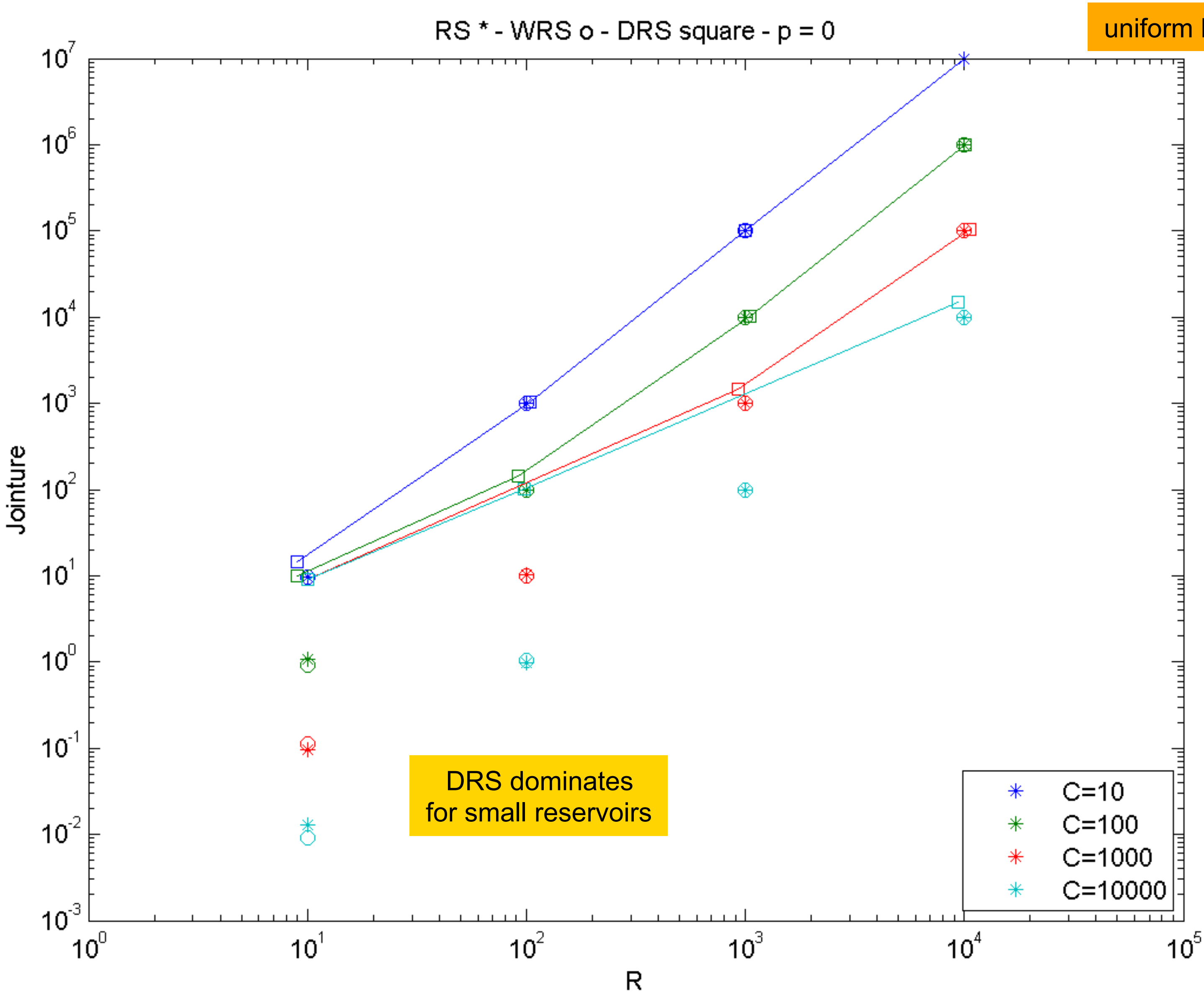
- keys in {0, …, C-1}

- controlled key distributions in streams 1 and 2
  - $P_1(k) \sim 10^{-p*k}$
  - $P_2(k) \sim 10^{-p*[(C-1) - k]}$

- from p=0 (uniform distributions in both streams) to p=4 (very unbalanced distributions)
- from C=10 to $C=10^4$

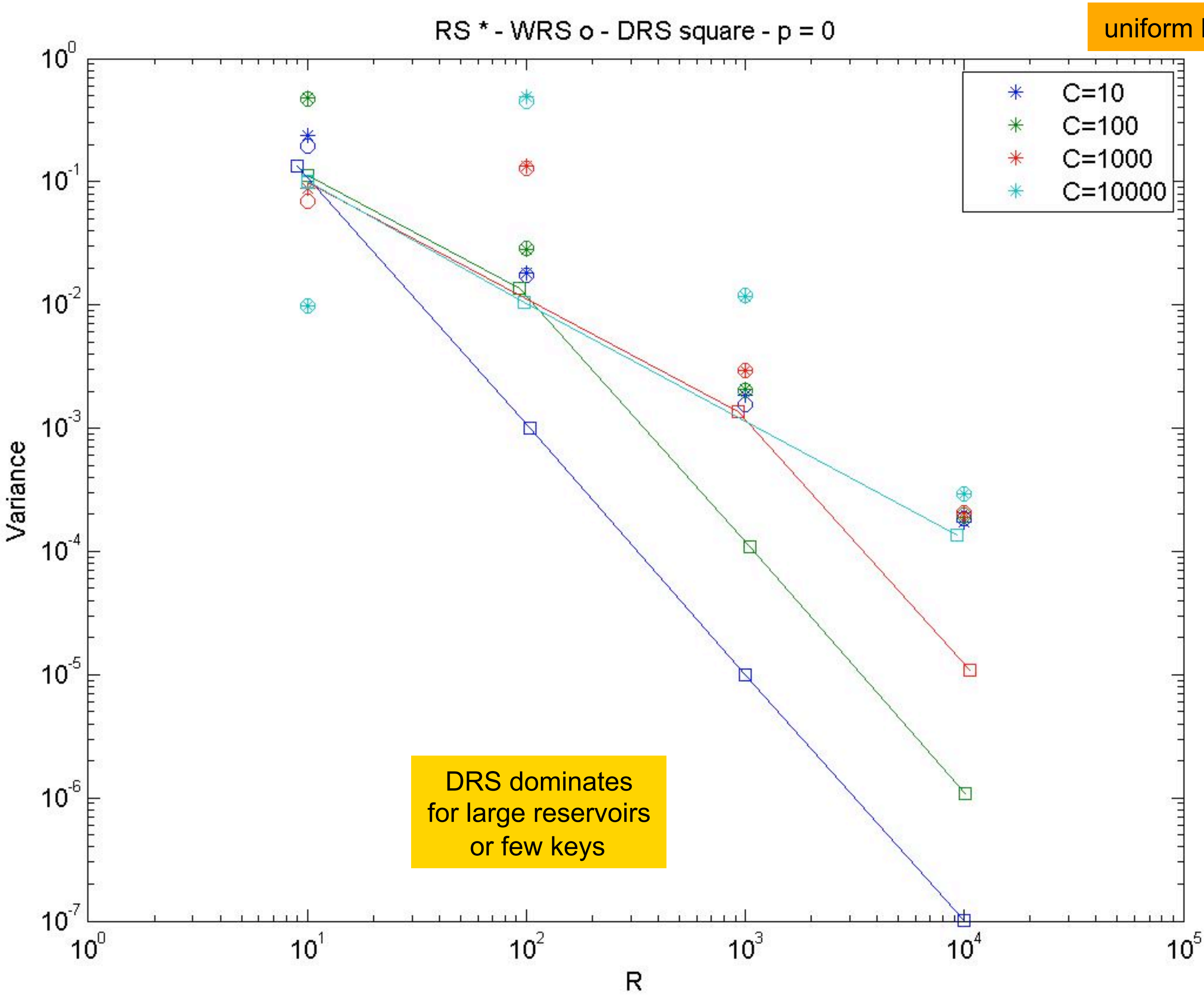
join size

uniform key distribution



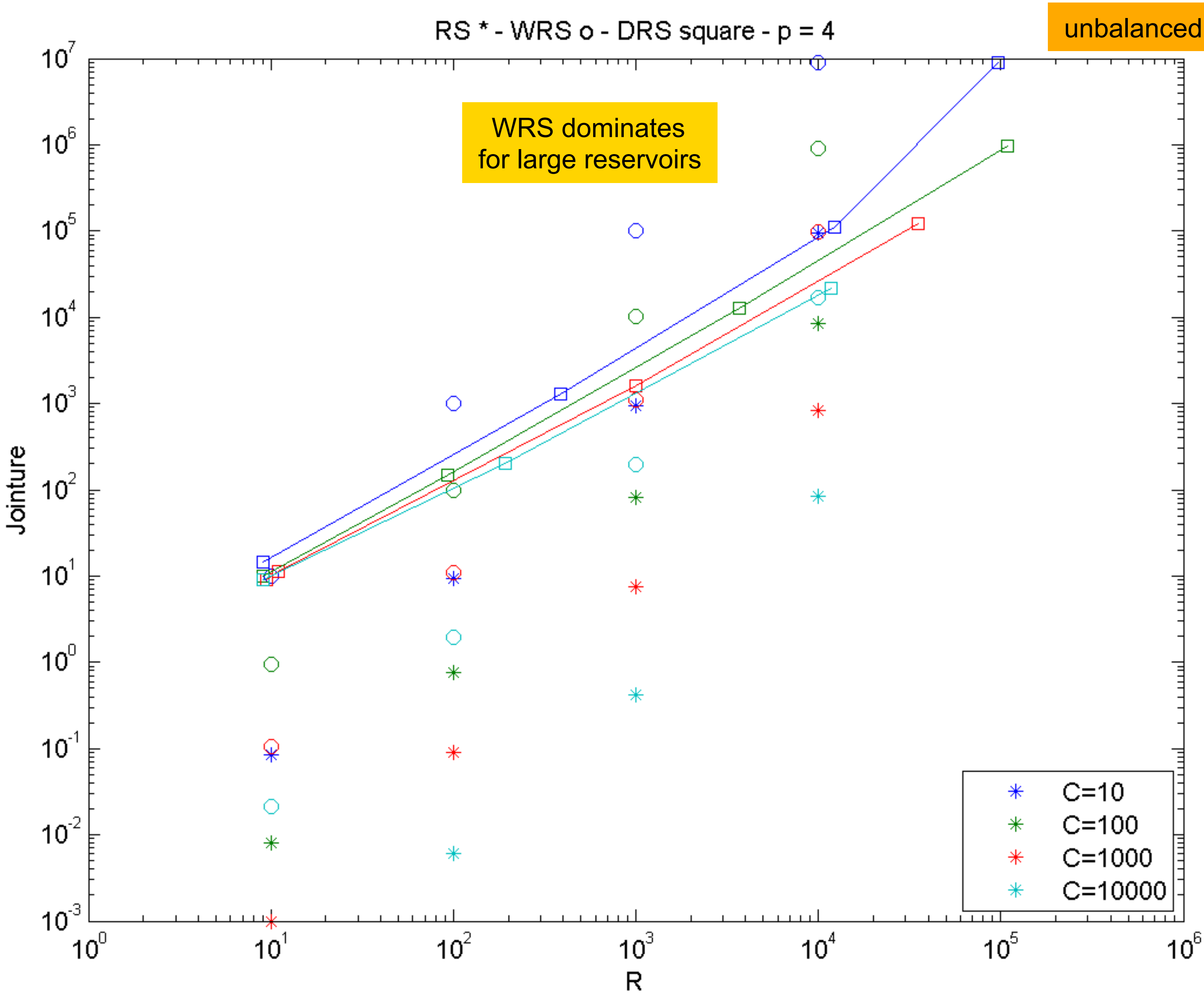
variance

uniform key distribution



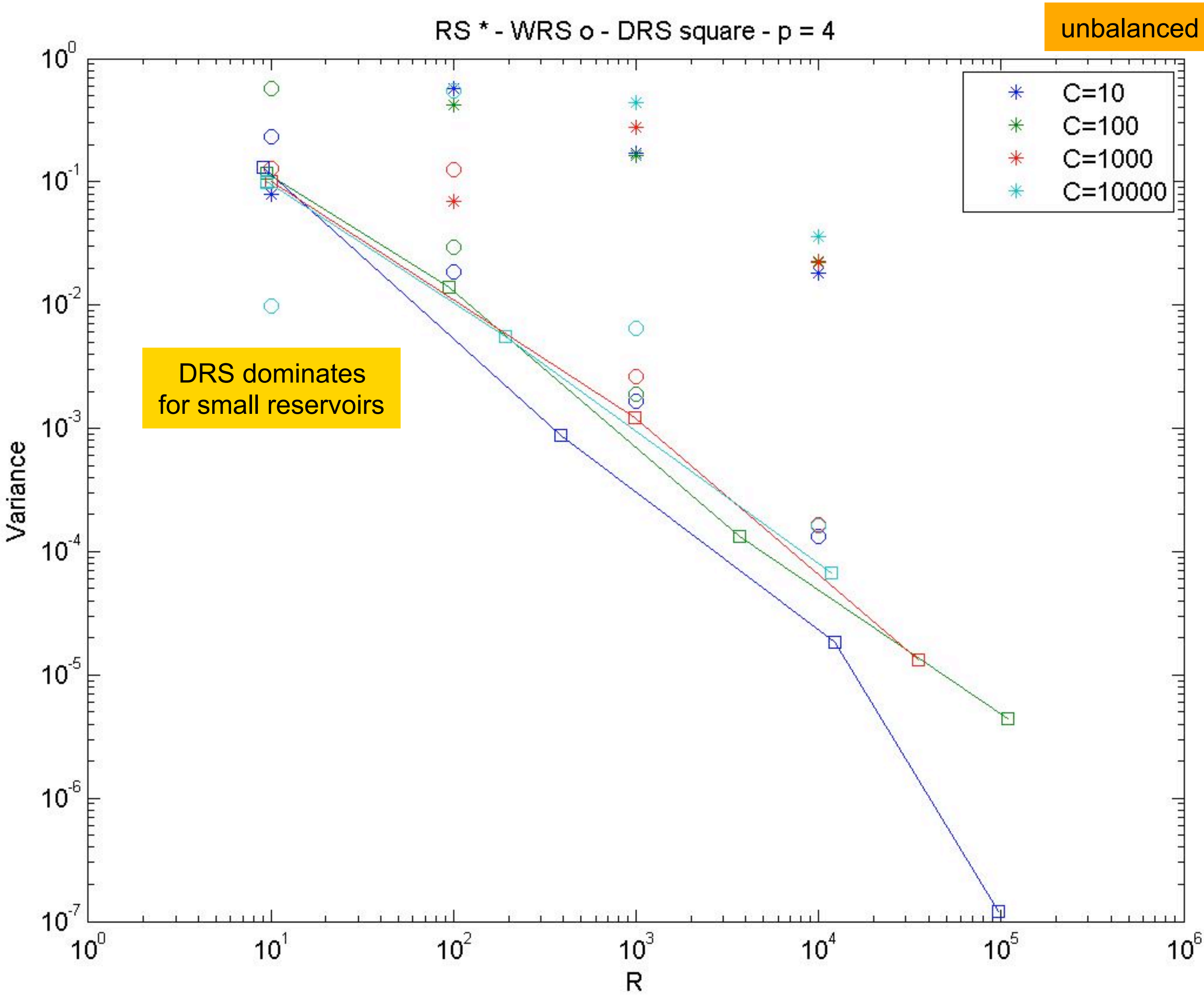
join size

unbalanced key distribution



variance

unbalanced key distribution



DRS dominates
for small reservoirs

second toy problem: conclusion

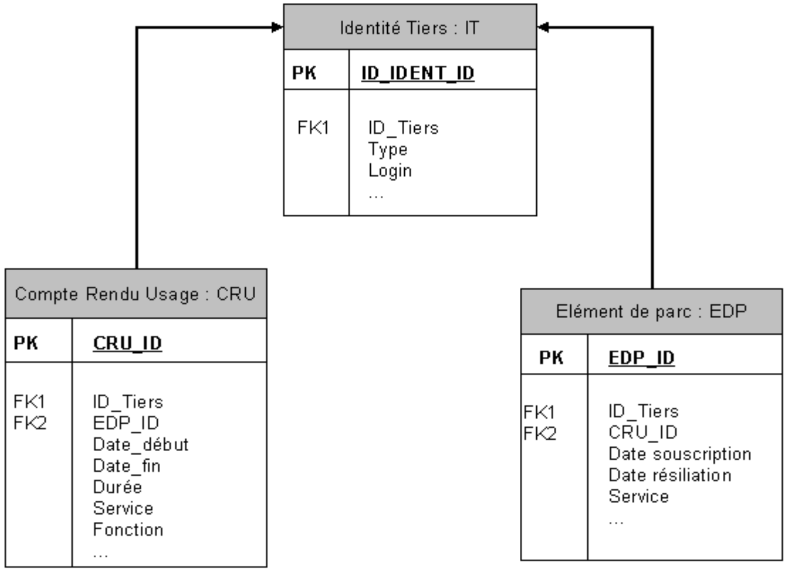
- variance: DRS dominates WRS
  - the simpler the problem (small number of keys, not too unbalanced distributions), the larger the domination at a given reservoir size
  - on complex problems, it takes very large reservoir sizes to get a noticeable domination of DRS on WRS
- join size: WRS makes better use of large reservoirs
  - WRS dominates for few keys and biased distributions
  - DRS dominates for small reservoirs

- WRS: complex problems AND large reservoirs
- DRS: simple problems OR small reservoirs

a real dataset

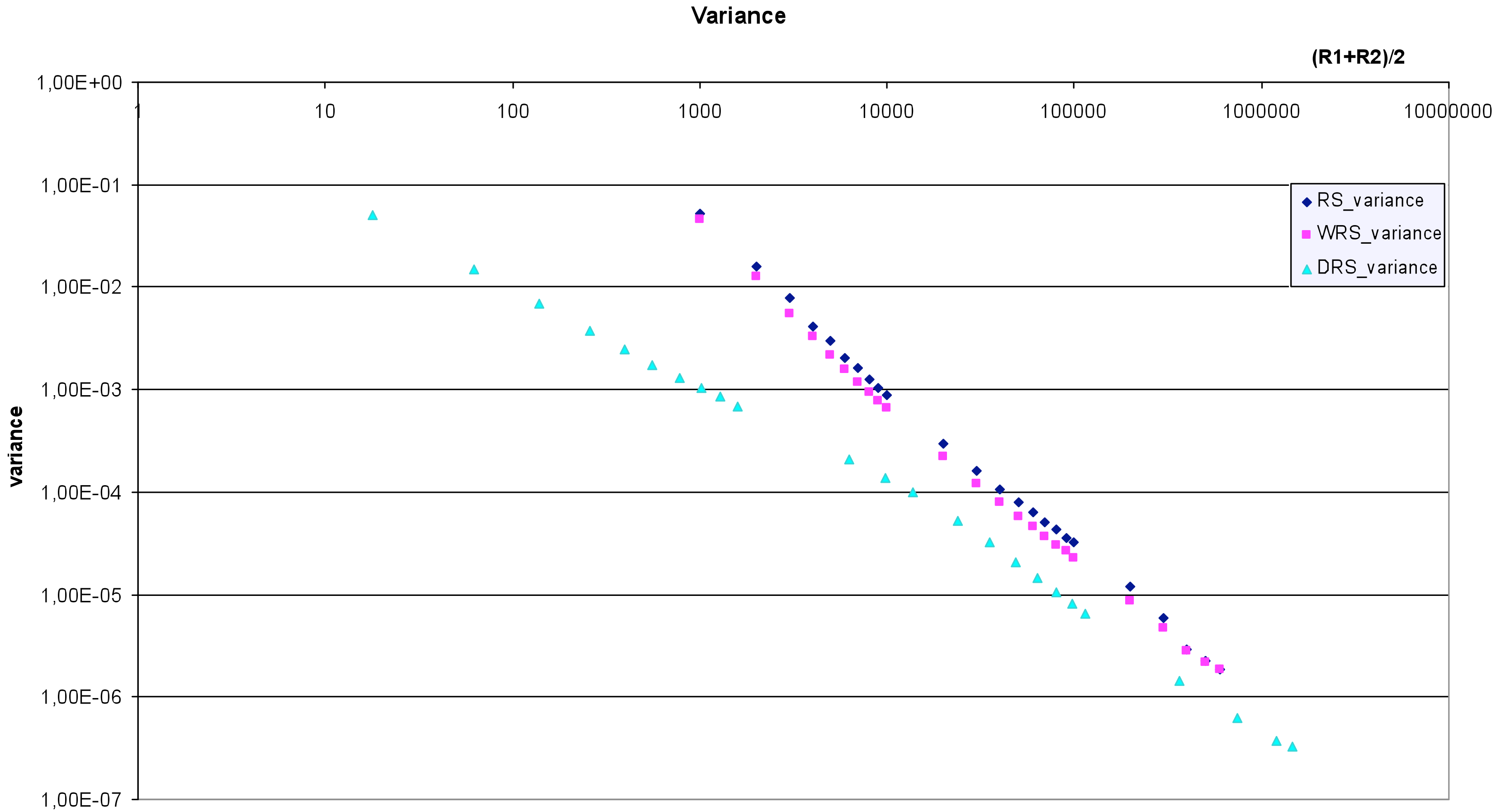
a large trace

- two traces are extracted from :
  - a services subscription: trace F1,
  - a use of services: trace F2.
- the join key is the user id.



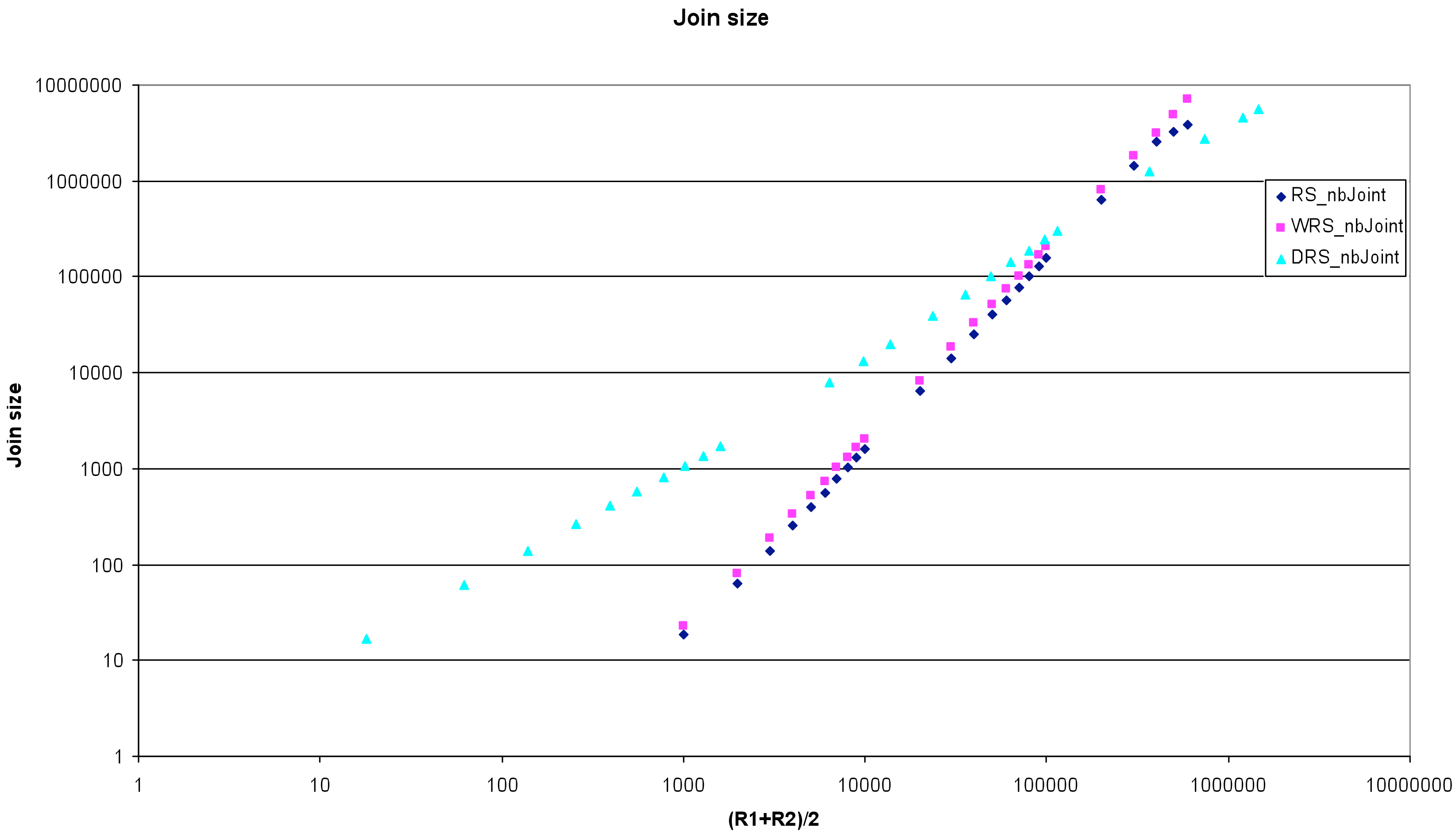
	F1	F2	F1><F2
size	408 333	30 239 496	19 550 134 962
# joint key	74 117	61 381	61 357

variance



Variance
(R1+R2)/2
1,00E+00
1,00E-01
1,00E-02
1,00E-03
1,00E-04
1,00E-05
1,00E-06
1,00E-07
1
10
100
1000
10000
100000
1000000
10000000
variance
RS_variance
WRS_variance
DRS_variance

join size



Join size
10000000
1000000
100000
10000
1000
100
10
1
1
10
100
1000
10000
100000
1000000
10000000
Join size
(R1+R2)/2
RS_nbJoint
WRS_nbJoint
DRS_nbJoint

large trace: conclusion

- with a large number of key Active Reservoir Sampling cannot be used (computational time cost n^3)

- Reservoir Sampling and Weighted Reservoir Sampling have almost equivalent performances
  - not too unbalanced key distribution in this example

- Deterministic Reservoir Sampling estimator outperforms other estimators for limited ressources.

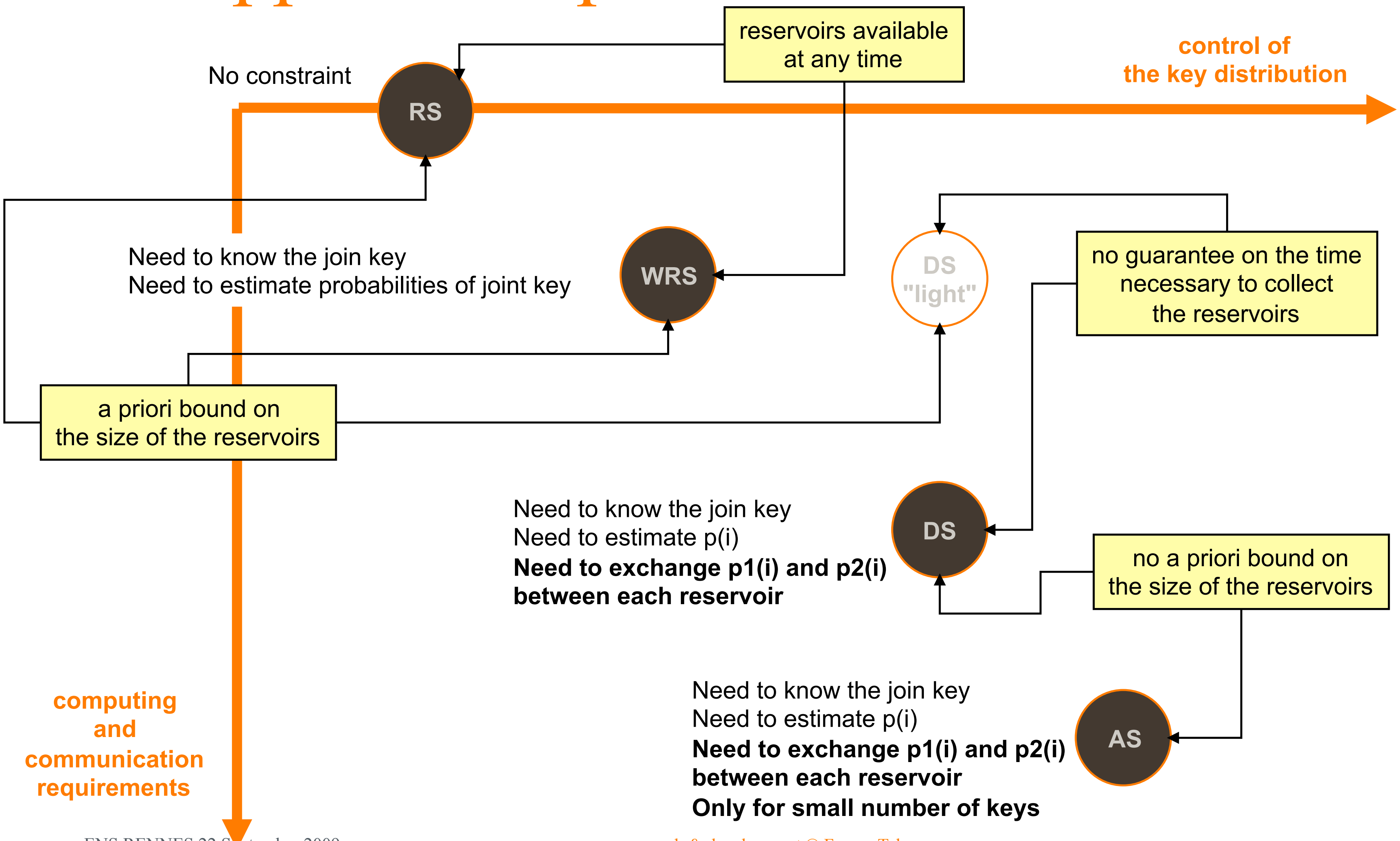
outline

- problem
- four possible methods
- experiments & results
- conclusion

so what … WRS or DRS ?

- well … sorry pal !
  - complex behaviours
  - no single best solution
    - WRS: complex problems AND large reservoirs
    - DRS: simple problems OR small reservoirs
- do you mean this is an helpless mess ?
- well ... not really: consider the applicative constraints first !

the application point of view



the application point of view