Diffuser du direct en pair-à-pair

Epidemic P2P live streaming

Thomas Bonald, Laurent Massoulié, Fabien Mathieu, Diego Perino, Andy Twigg

Orange Labs / Thomson Paris

10/10/08, **IRISATECH**

Outline

- P2P Live Streaming
  - A brief introduction
  - Optimal diffusion
  - The epidemic approach
- Epidemic schemes
  - A push, discrete, model
  - Schemes building
  - Schemes behaviors
- Performance
  - Main metrics
  - Theoretical results
  - Simulations
  - Summary
- Conclusion

P2P Live Streaming

One to give, many to watch

What is Live?

VoD

Live

Video-on-Demand		Live
Anywhat anywhen	Specific QoS	play-out delay
Content known in advance	Technical pro	Single users' behavior
Multiple users' behaviors	Technical challenge	Content created on the fly

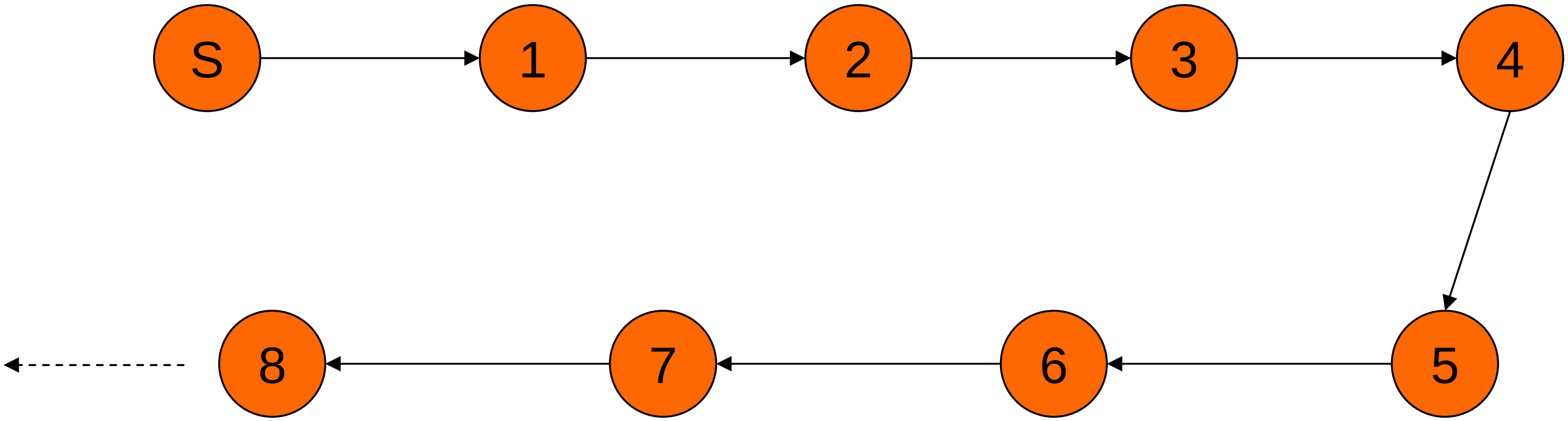
P2P?

- Scenario: a limited source injects live content into a network
  - Live ! cannot be pre-fetched
  - Limited ! not enough bandwidth to sustain the demand
- Watchers (peers) with upload capacities want the content
- Goal: use watchers' capacities to deliver the content
  - As fast as possible (small play-out delay)
  - With good quality (rate/losses)

Case Study

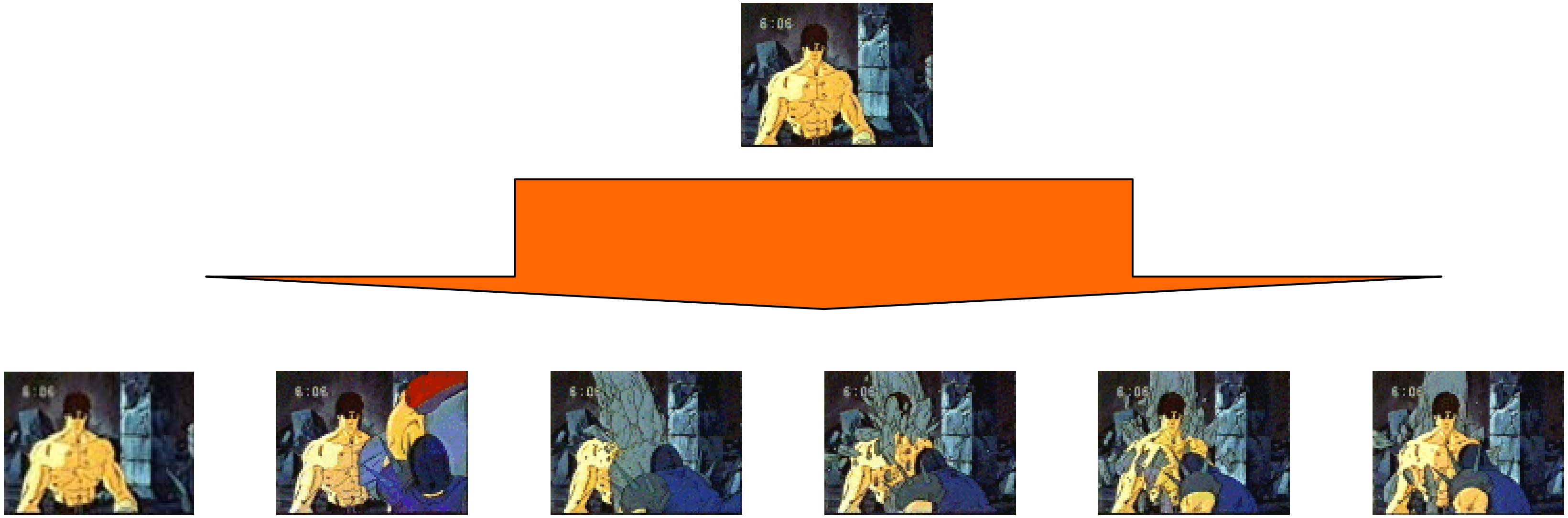
- The source injects the content
  - Continuous stream of undefined length
  - Only one copy
- Fixed number *n* of watchers (peers)
  - Each one wants the stream
  - Each one is able to relay one copy of the stream it receives
- How can things work?

Linear diffusion



- All peers can watch the stream, but…
- So long for the play-out!
- A single peer's malfunction blinds half the swarm (in average)

Workaround: splitting into chunks

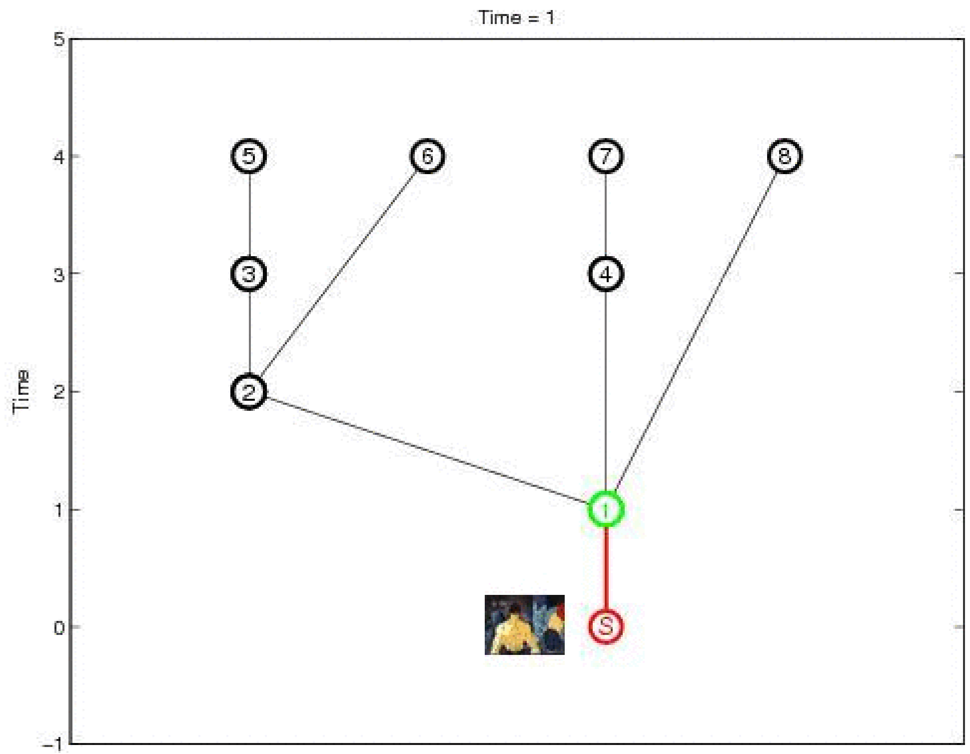


- The content is divided into pieces of data (chunks),
- The source injects one new chunk per time unit (round)
- Peers only relay full chunks they have

Can it work better?

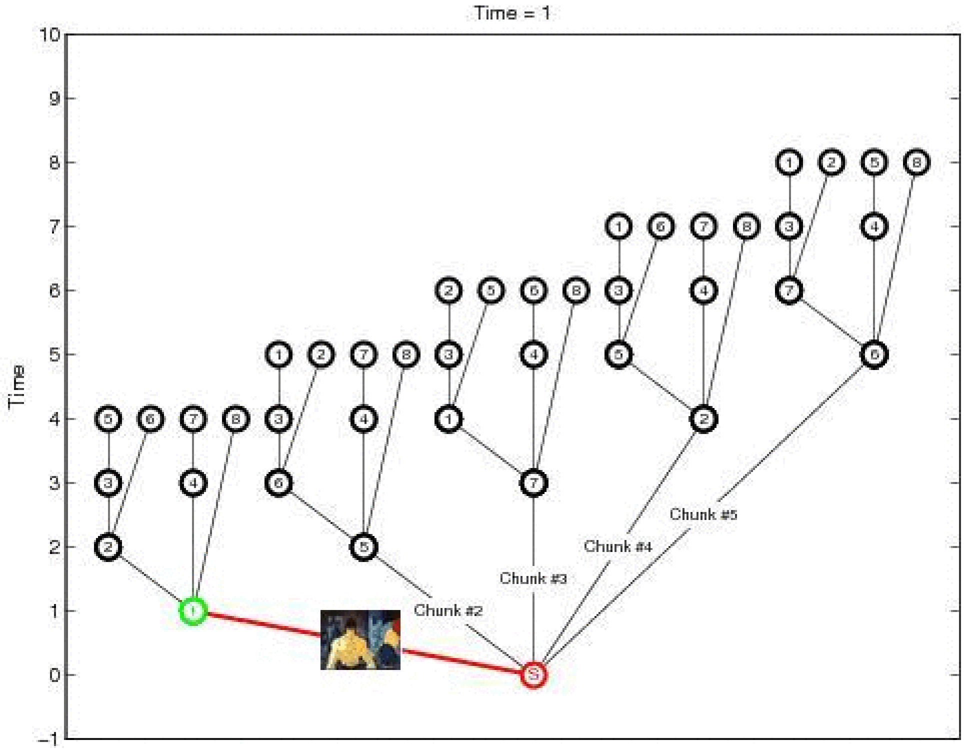
Single chunk distribution

- One peer receive the chunk (*t=1*)
- As long as necessary, peers with the chunk upload to peers without it
- It is optimal:
  - chunk delivered to all peers in log(n)
  - A single peer's malfunction only affects log(n) peers (in average)



Multiple chunks distribution

- If we want optimality, each diffusion tree must be a permutation of the single chunk optimal tree
- Potential download/upload conflicts:
  - A node cannot be red (uploading) for two chunks at the same time
  - If download bandwidth is constrained, same for green
- It can be done!



Multiple, interleaved, optimal trees are structured

- They are nice:
  - Resources are not wasted
  - Delay is optimal
  - Things can be proved
- But…
  - They require a tight scheduling
  - Full knowledge is required to set up the scheduling
  - Churning/failures require full rescheduling
  - What if heterogeneous capacities?

Going epidemic (unstructured)

- Introduce random decentralized choices
- Use *luck* to achieve diffusion and avoid conflicts
- More flexible, but maybe less optimal
  - How far from the optimal?
  - What do we gain?

Building epidemic schemes

Push and you shall diffuse

Understanding how unstructured differs from stucture

- Structured is about… structures!
  - You have seen an example in this talk…
  - And many others in the previous ones!
- Unstructrured is about… behaviors!
  - You tell peers how to behave
  - You let things happen
  - Does not mean there is no structure (but it's not explicit)

Going epidemic: the push approach

- In real systems, diffusion is a combination of
  - Requests for download (pull)
  - Selections for upload (push)
- The push model only considers the uploader strategy
- Epidemic: the behavior should aim at chunks virus-like propagation
- Tells the peers how to use their upload (push)
- Suitable for upload-limited systems
- All the strategy is embedded in a *push scheme function*

Push schemes

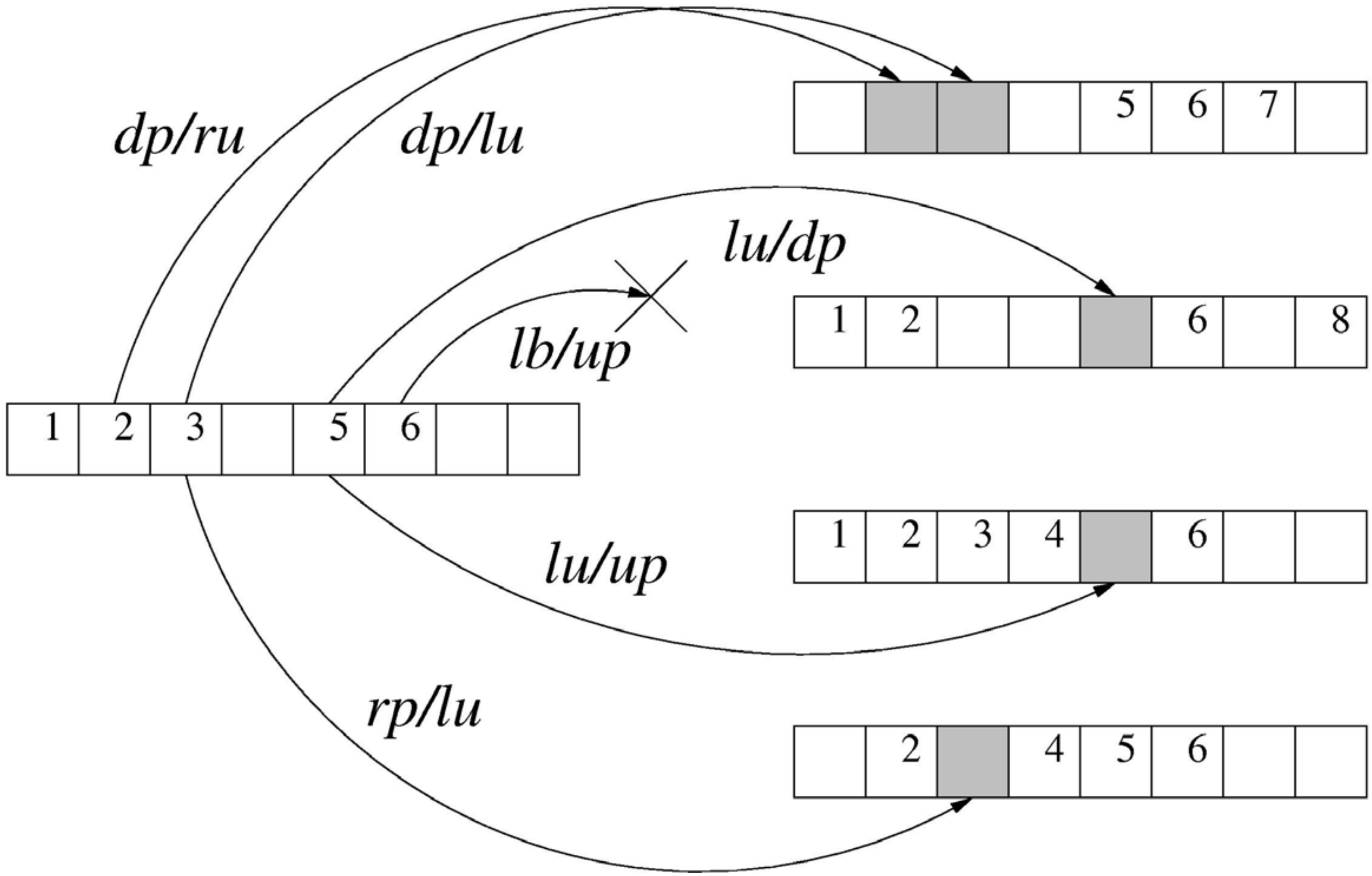
- A push scheme tells peers where to push which chunk:
- Input:
  – a peer ready to upload a chunk,
  – Additional knowledge (optional)
- Output:
  – a destination peer
  – a chunk Id
- Most (but not all) basic schemes fall into one of these:
  – Select destination, then select chunk
  – Select chunk, then select destination

Basic Push Schemes

- Destination selection:
  - Random (rp)
  - Random useful (up)
  - Most deprived (dp)
- Chunk selection:
  - Latest blind (lb)
  - Latest useful (lu)
  - Random useful (ru)
- You build schemes lego-style

! rp/lb(=lb/rp), rp/lu, dp/lu, dp/ru, lb/up, lu/up, lu/dp

Schemes behaviors

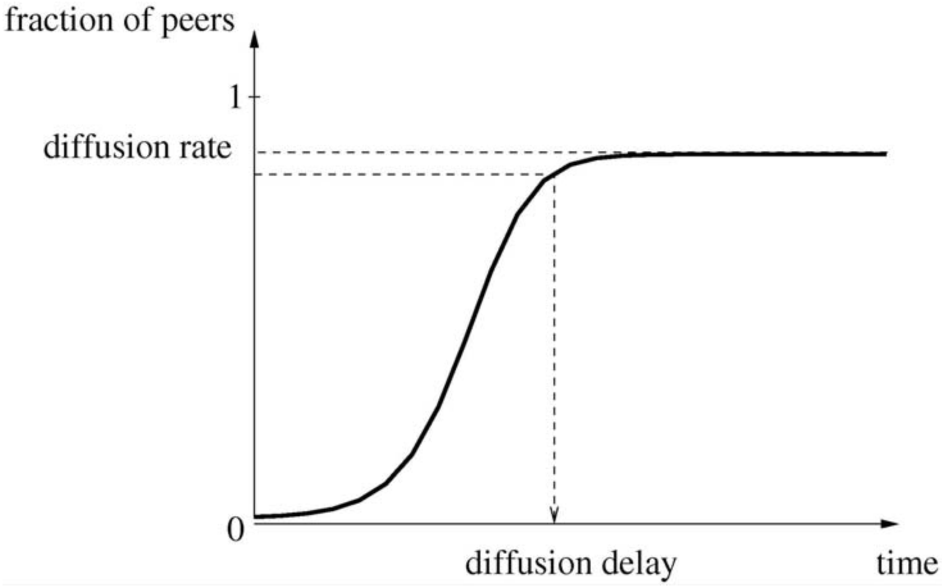


Performance

Rate is nothing without delay

Main performance metrics: rate and delay

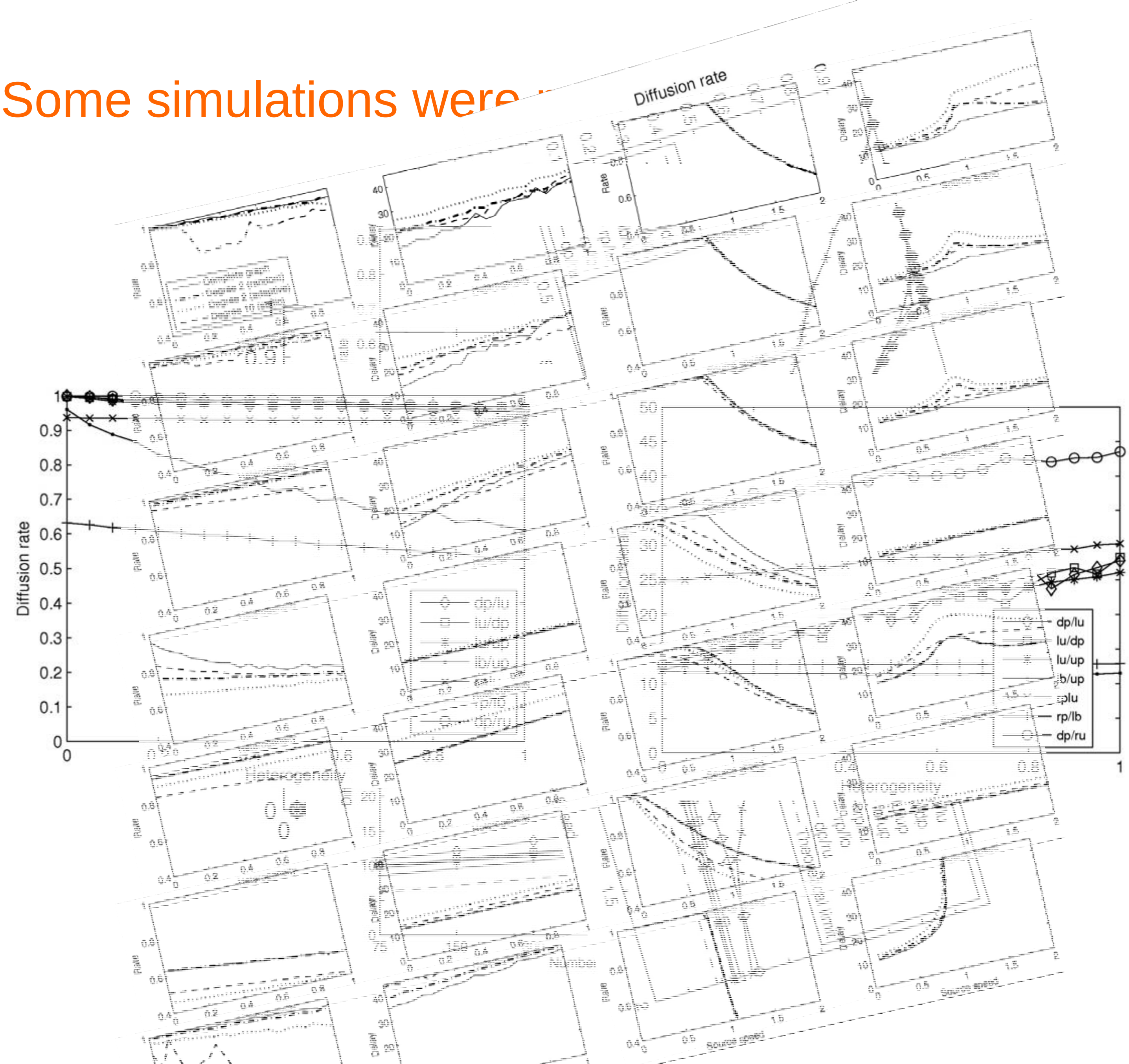
- All epidemic schemes lead to S-shaped diffusion
  Price of anarchy
- Diffusion rate *r(t)*: average diffusion of a chunk after a given delay *t*
  What is the asymtotic rate?
- Diffusion delay *d(r)*: average delay needed for achieving a given rate *r*
  What delay to achieve almost the asymptotic rate?
- Other metrics:
  - Overhead
  - Robustness
  - Cheat-proof-ness
  - …



Theoretical results

- Things can be proved for epidemic schemes (but it's harder than for structured solutions)
- For rp/lu (random peer latest useful): with just a little bit extra-bandwidth capacity, it can achieve full rate in optimal delay with high probability, *up to a constant*
- For some other schemes, we can give formulas for the S-shaped curves
- But when you cannot use theory, you have to switch to simulations

Some simulations were



Schemes performance (summary)

Scheme	Optimality?	H. impact	Overhead	Cheat?
rp/lb (=lb/rp)	Delay optimal (proved) Rate sub-optimal (lot of collisions)	Rate (-15%)	;	No
rp/lu	Optimal outside critical zone. Delay constant can be huge	Delay (+20%)	Light	No
lb/up	Delay optimal Rate optimal outside critical zone. Optimal as n!1	Rate (-35%)	Huge	No
dp/ru	Rate optimal (proved) Unstable delay (overloaded regimes)	Rate (-5%) Delay (+20%)	Huger Can be reduced (power of 2 choices)	No (lu/up) Yes (*/dp/*)
lu/up lu/dp dp/lu	Optimal	Rate (-5%) Delay (+100%)		

Conclusion/Future work

- We have shown simple unstructured epidemic schemes can perform almost as good as complex and rigid structured schemes
- No true winner, but open the path to next generation schemes
- Future work may consist in
  - Proving optimality for more schemes
  - Refining the model for realistic continuous speeds
  - Adapting the simulator engine
  - Find true winner schemes (almost done)

Merci beaucoup !

Des questions ?