SSSim: a Simple and Scalable Simulator for P2P Streaming Systems

CAMAD '09

Luca Abeni, Csaba Kiraly, and Renato Lo Cigno

Overview of the Talk

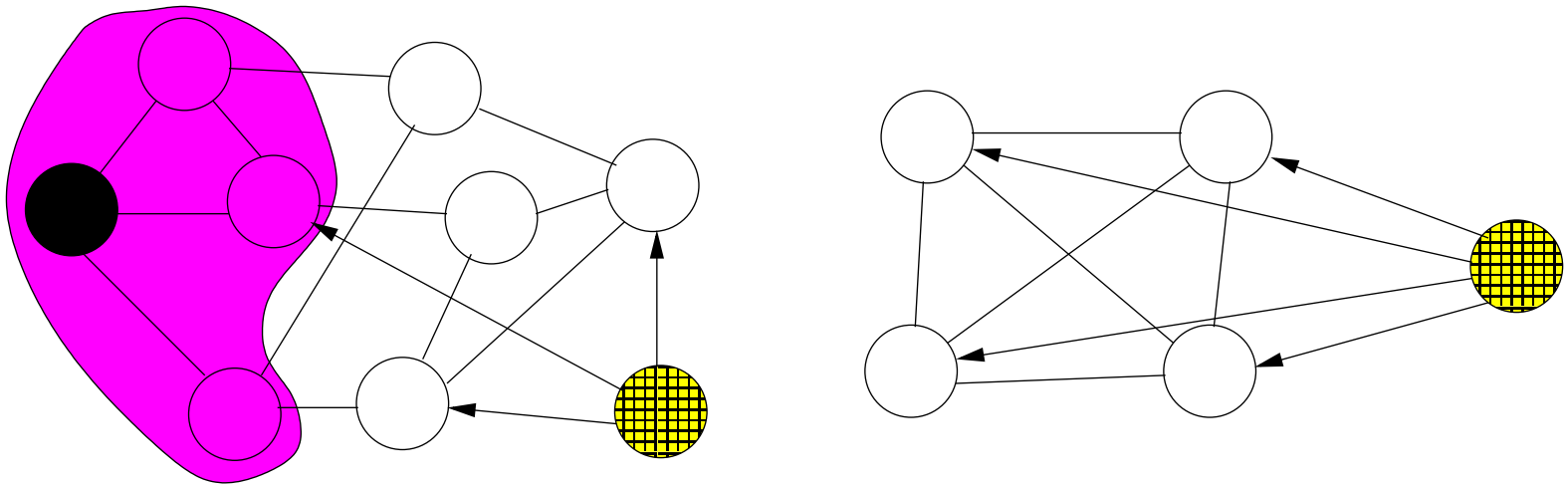
- Introduction / Definitions
  - P2P Streaming Systems
  - Unstructured systems / Neighbourhood / ...
  - Schedulers
- Simulator Requirements
  - Simplicity: implementing new features/schedulers should be easy
  - Scalability: to simulate large amounts of peers, long streams, large neighbourhoods, ...
- SSSim
  - Modular and flexible
- Some experimental results

P2P Streaming Systems

- A Source generates encoded audio/video
- Various Peers receive the encoded media and contribute to the diffusion (by forwarding the received media)
- The encoded media is characterised by *implicit temporal constraints*
- The perceived QoS depends on how the constraints are respected
  - Affected by the decisions taken by each peer (schedulers)
  - Performance evaluation (diffusion delay, ...) $\rightarrow$ simulations

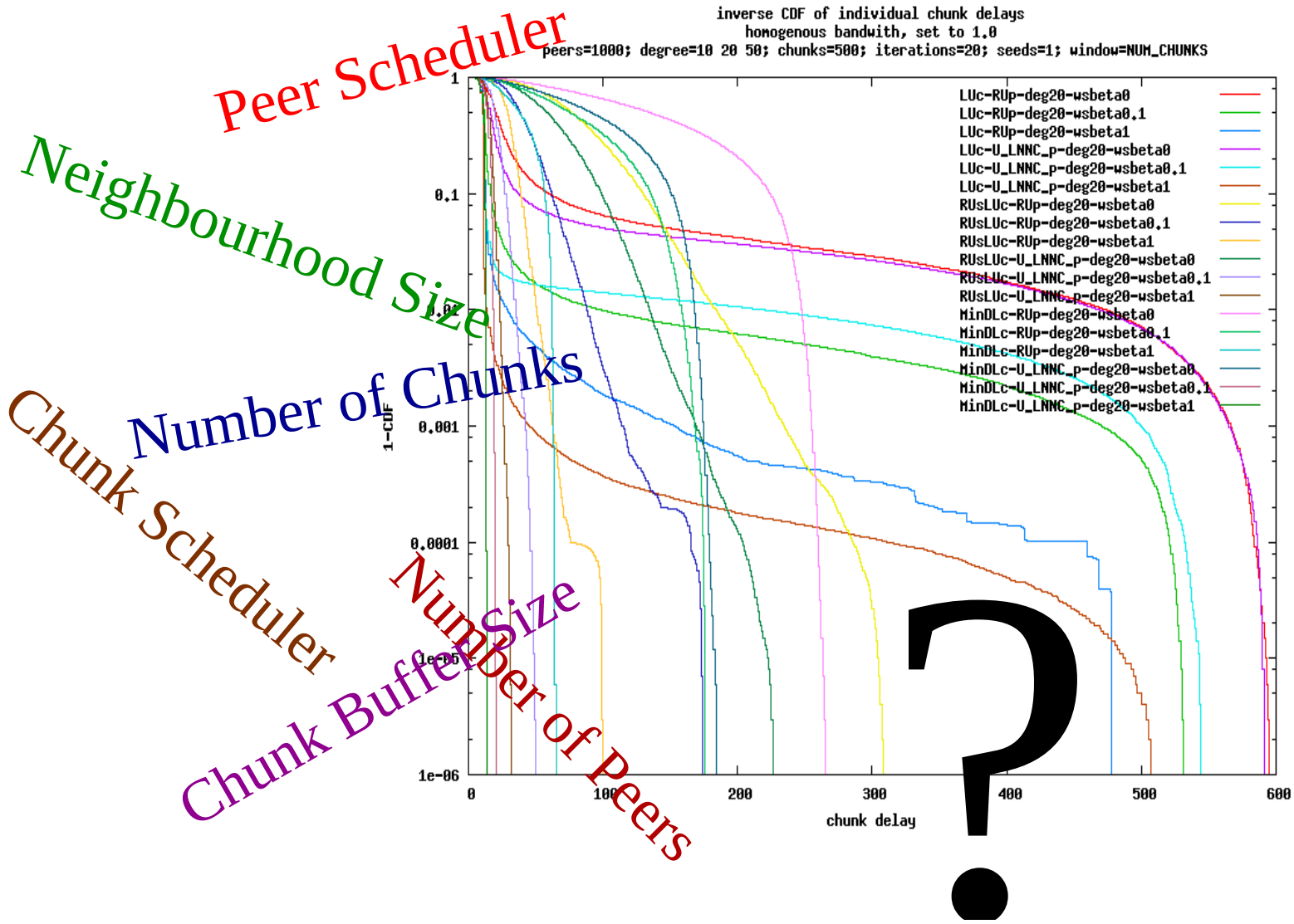
Definitions

- The media stream is divided in chunks
- Unstructured system



- Each peer is connected to a subset of the other peers (neighbourhood)
  - Source → only sends chunks (directed links)
  - Other peers → bidirectional links

Performance Evaluation - 1



Performance Evaluation - 2

- Need to run large numbers of simulations
  - Different parameters (peer number, neighbourhood size, chunk number, ...)
  - Different scheduling algorithms
  - Multiple runs
- Need to modify scheduling algorithms/modify the current ones
- Need to modify the simulator/adapt it to new requirements

Requirements

- Performance (memory and CPU time)
- Simplicity
  - Easy to hack, and to adapt to the users' needs
- Scalability
  - CPU time scalability is not enough... Memory scalability is needed! (avoid thrashing, etc...)
- Possibility to easily add/implement new scheduling algorithms
- Sometimes, these requirements are more important than having a completely general-purpose simulator
  - Specialized simulator: discrete-time (instead of event-based)

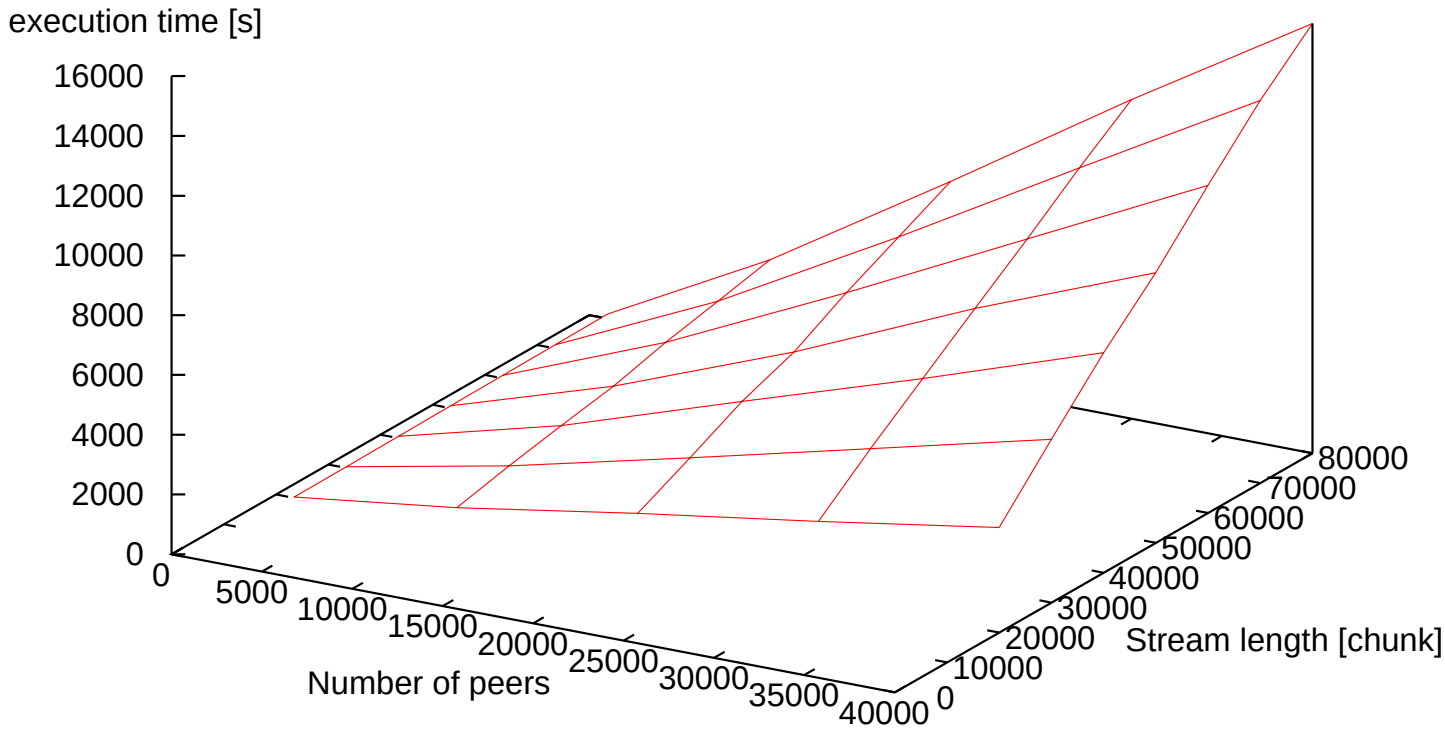
SSSim

- Not event based (avoid the event-queue overhead)
- Modular:
  1. Overlay generation module
     - Easily customisable/configurable
     - Import/Export graphviz files
  2. Scheduler
     - `schedule(struct peer *p, int *chunk, struct peer **target)`
     - Helpers and macros for defining schedulers
  3. Main loop
  4. Statistics
- Optimised for the important cases
- Currently assumes that each peer sends one chunk per cycle

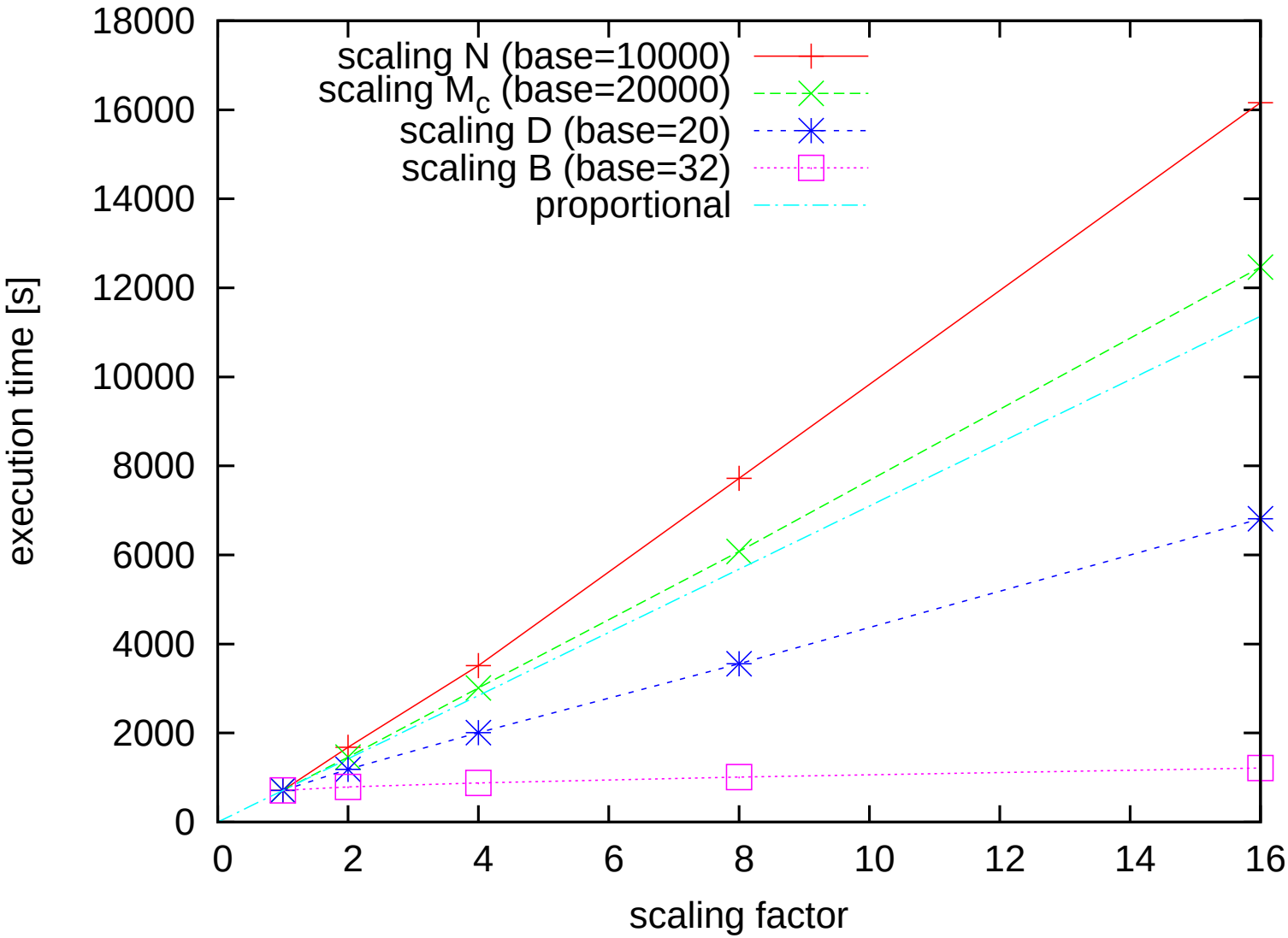
Optimisations: Some Examples

- If every copy of every chunk is stored in memory:
  - $N = 10000$ peers, $M_c = 20000$ chunks $\Rightarrow 1.5GB$ of memory, if a chunk is 8 bytes
  - High memory pressure, slowdown, risk of thrashing, cache problems...
- Solution: only store the latest B chunks in memory (chunk buffer)
- Chunk buffer handling $\rightarrow$ overhead
  - Optimisation: chunk number i goes in position $i\%B$ of the buffer
  - Computing $i\%B$ requires a division: slow!!!
  - If $B = 2^k$, $i\%B = i\&(B-1)$: 5 times faster!!!

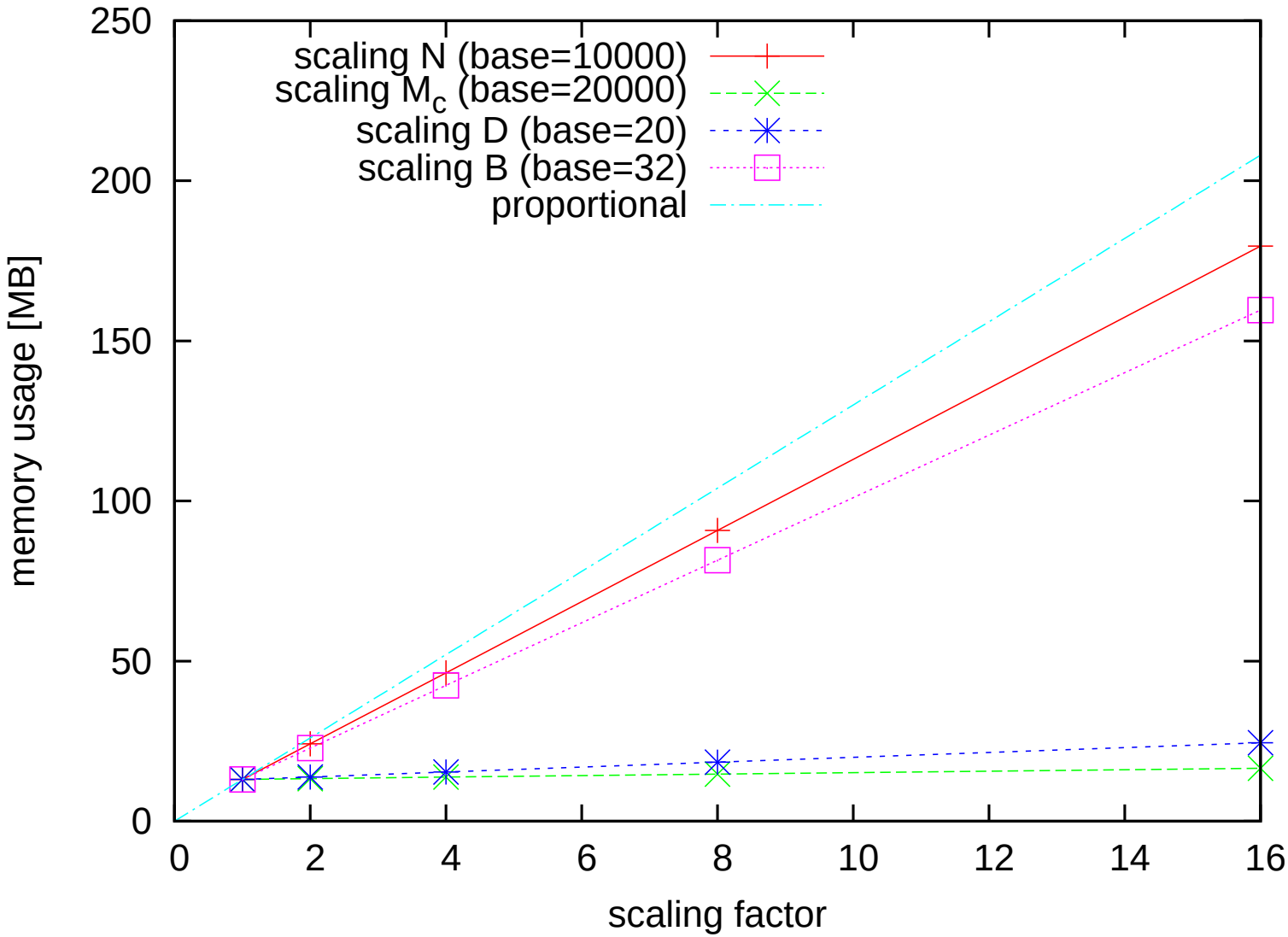
CPU Time Scalability - 1



CPU Time Scalability - 2



Memory Scalability



Comparisons

Comparisons between SSSim and a more generic (event-based) simulator (P2PTVSim)

- Much better memory scalability (with the number of peers N)
- Much better CPU time scalability with the buffer size B

N	M_c	D	B	SSSim Time	SSSim Memory	P2PTVSim Time	P2PTVSim Memory
1000	3000	999	32	$72s$	$2.945MB$	$835s$	$23.473MB$
2000	3000	999	32	$350s$	$3.9MB$	$5609s$	$82.461MB$
3000	3000	999	32	$993s$	$4.957MB$	$17355s$	$179.723MB$
1000	3000	999	3000	$371s$	$101.801MB$	$80282s$	$114MB$
10000	10000	20	32	$355s$	$12.953MB$	$2138s$	$21.148MB$

Conclusions

- SSSim: Simple and Scalable Simulator
- It is free too (“free as in freedom”, but also “free as in beer”)
  - Released under the GPL
- Check `http://imedia.disi.unitn.it/SSSim`
  - Download it, test it...
  - ...and feel free to contribute!