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, ...) → simulations
Definitions

- The media stream is divided in **chunks**
- **Unstructured system**

Each peer is connected to a subset of the other peers (neighbourhood)
- **Source** $\rightarrow$ only sends chunks (directed links)
- **Other peers** $\rightarrow$ bidirectional links
Performance Evaluation - 1

![Graph showing inverse CDF of individual chunk delays]

- Number of Peers
- Chunk Buffer Size
- Chunk Scheduler
- Neighbourhood Size
- Number of Chunks
- Peer Scheduler

SSSim: a Simple and Scalable Simulator for P2P Streaming Systems – p. 5/14
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 - 2

![Graph showing CPU time scalability with different scaling factors for parameters N, M_c, D, B.](image)
Memory Scalability

The diagram shows the memory usage [MB] plotted against the scaling factor. Various scaling factors are demonstrated, including scaling N (base=10000), scaling M_c (base=20000), scaling D (base=20), and scaling B (base=32). The graph also illustrates a proportional scaling line. The memory usage increases linearly with the scaling factor.
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 $B$

<table>
<thead>
<tr>
<th>$N$</th>
<th>$M_c$</th>
<th>$D$</th>
<th>$B$</th>
<th>SSSim Time</th>
<th>SSSim Memory</th>
<th>P2PTVSim Time</th>
<th>P2PTVSim Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>3000</td>
<td>999</td>
<td>32</td>
<td>72s</td>
<td>2.945MB</td>
<td>835s</td>
<td>23.473MB</td>
</tr>
<tr>
<td>2000</td>
<td>3000</td>
<td>999</td>
<td>32</td>
<td>350s</td>
<td>3.9MB</td>
<td>5609s</td>
<td>82.461MB</td>
</tr>
<tr>
<td>3000</td>
<td>3000</td>
<td>999</td>
<td>32</td>
<td>993s</td>
<td>4.957MB</td>
<td>17355s</td>
<td>179.723MB</td>
</tr>
<tr>
<td>1000</td>
<td>3000</td>
<td>999</td>
<td>3000</td>
<td>371s</td>
<td>101.801MB</td>
<td>80282s</td>
<td>114MB</td>
</tr>
<tr>
<td>10000</td>
<td>10000</td>
<td>20</td>
<td>32</td>
<td>355s</td>
<td>12.953MB</td>
<td>2138s</td>
<td>21.148MB</td>
</tr>
</tbody>
</table>
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](http://imedia.disi.unitn.it/SSSim)
  - Download it, test it...
  - ...and feel **free** to contribute!