Adaptive Video Streaming

A Survey and Case Study

HU, Pili

http://personal.ie.cuhk.edu.hk/~hpl011/

December 19, 2011

Landscape of Adaptive Video Streaming

- MDC, MLC.
- Multicast(1996), Unicast(2001), P2P(still chaos)
- ASTRI:
  - Current Version: multi channel layered VoD (real deployment)
  - Simulation platform: single channel fixed initial layer VoD (trial version, [wen, 2010])

Landscape of Adaptive Video Streaming

Version Trees

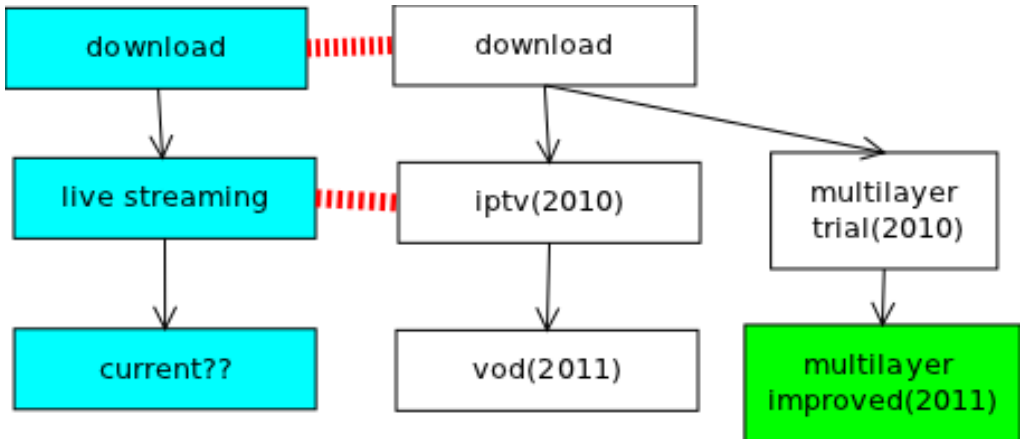
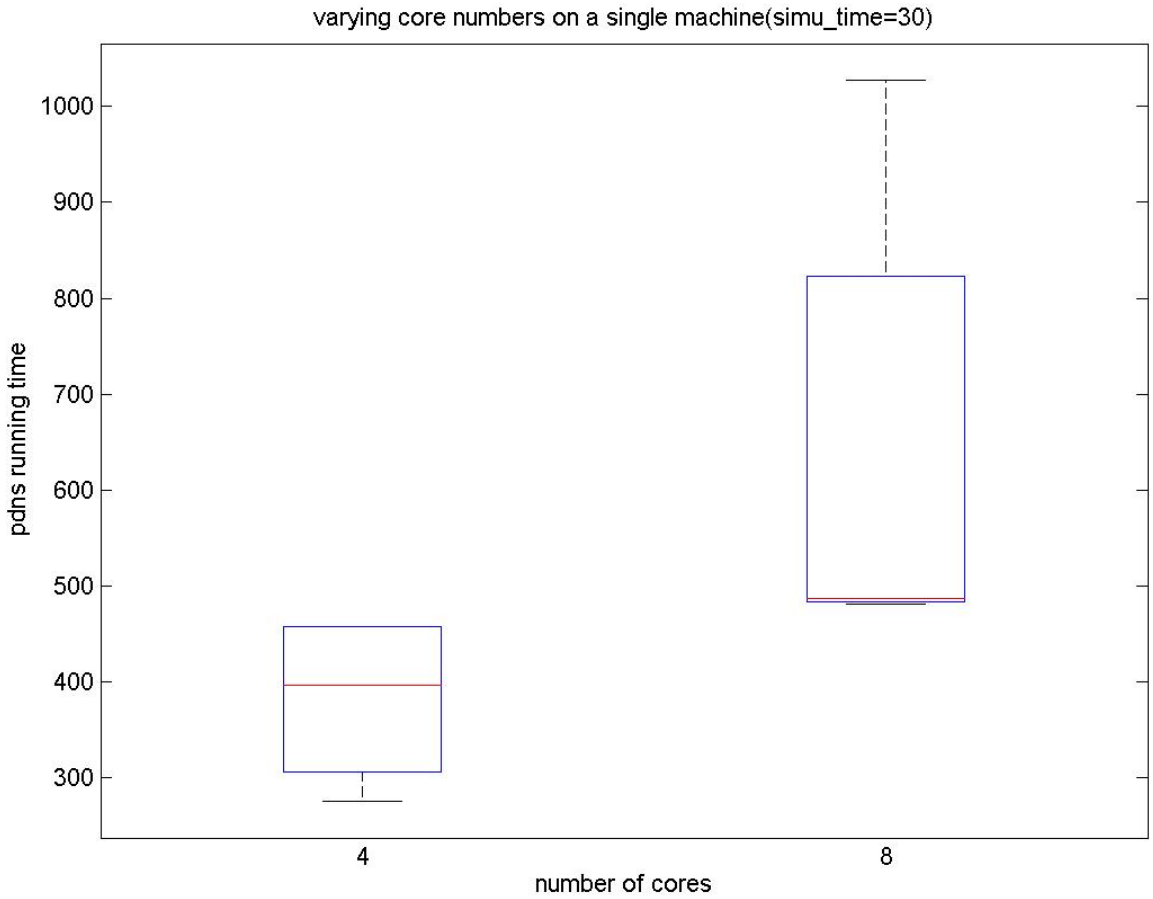


Figure: Version Tree of Real and Simulation

- Blue: Real deployment
- While: Simulation platform
- Green: This work
- Red dash: Equivalency

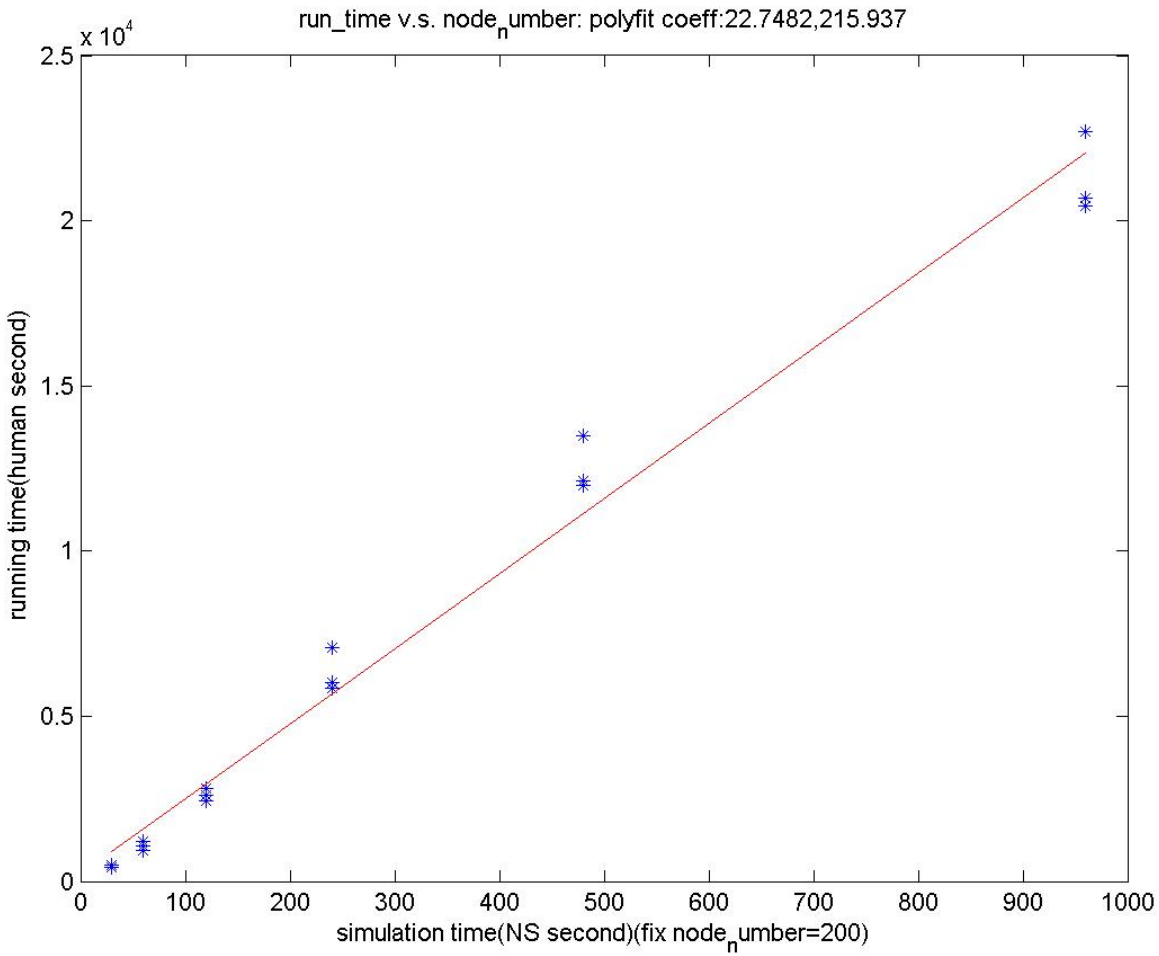
Performance Benchmark

runtime v.s. # of core



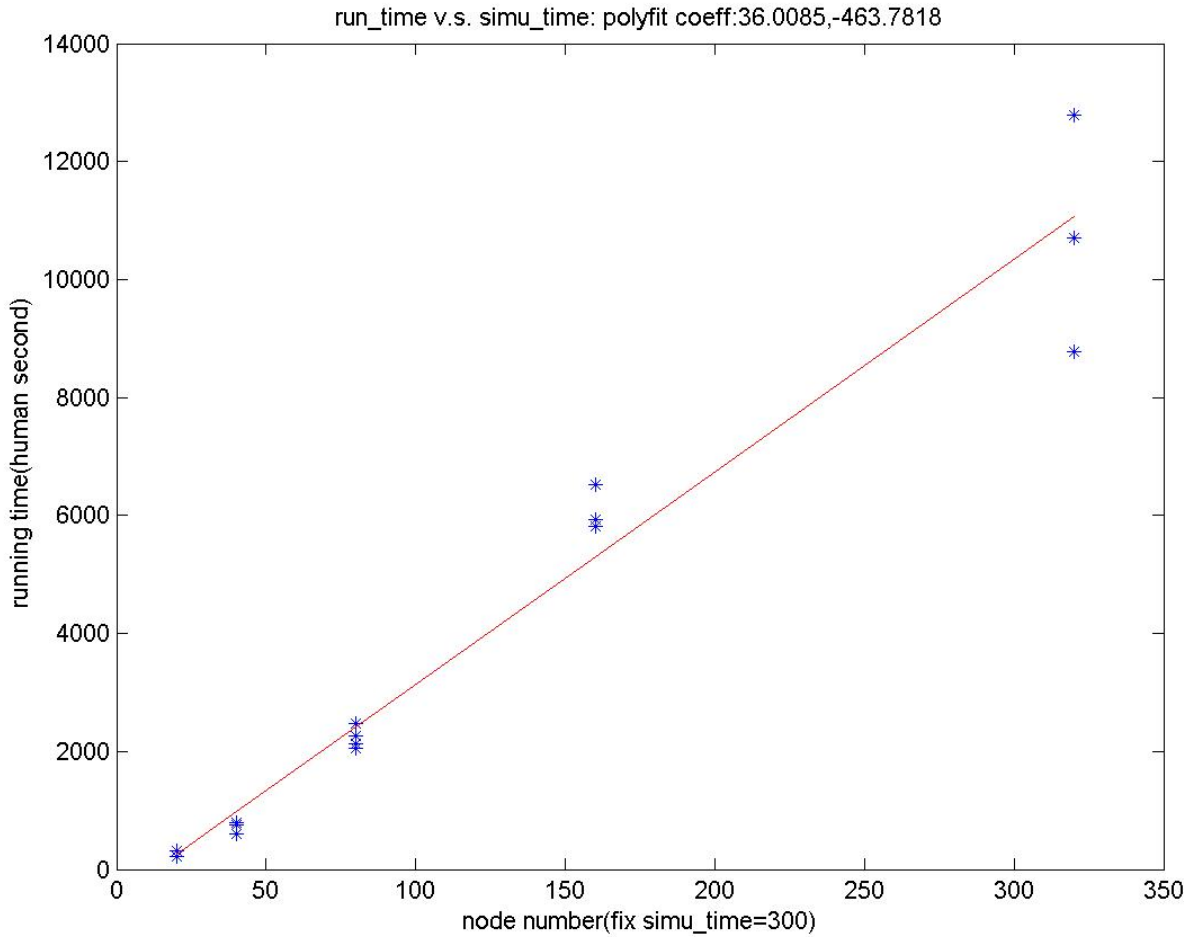
Performance Benchmark

runtime v.s. simulation time



Performance Benchmark

runtime v.s. # of nodes



Simulation Confuguration

Basic Settings

- Number of Nodes: 160
- Simulation Time: 300 (NS seconds)

Simulation Confuguration

Topology and Capacity

Wen Zheng's configuration:

- Star like, 4 subnet.
- Subnet parameters:
  - Subnet1: down:10Mb; up:10Mb.
  - Subnet1: down:3Mb; up:0.5Mb.
  - Subnet1: down:3Mb; up:0.5Mb.
  - Subnet1: down:3Mb; up:0.5Mb.

Simulation Confuguration

Layers

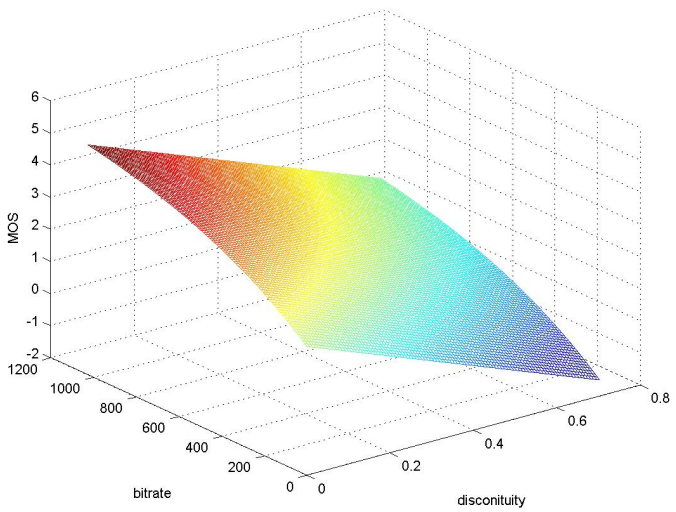
- 3 Layers. Conform to [wang,2011] QoE study.
- Layer parameters:
  - Layer1: 256Kbit. (per piece)
  - Layer2: 256Kbit. (per piece)
  - Layer3: 512Kbit. (per piece)

Cumulative layer piece size is: 256, 512, 1024 (Kbit).

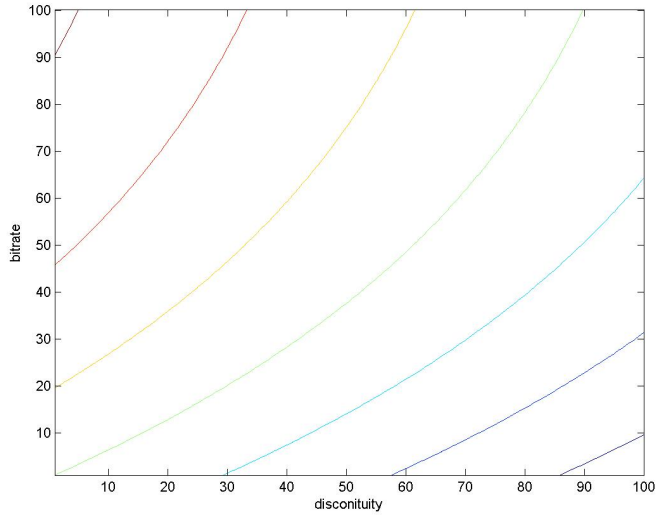
QoE Model Adopted

[wang, 2011], B-D tradeoff, continuous version.

$$MOS = c_1 \times d + \alpha \times (1 - e^{-b \times \lambda}) + c_2 \quad (1)$$



(a) 3D

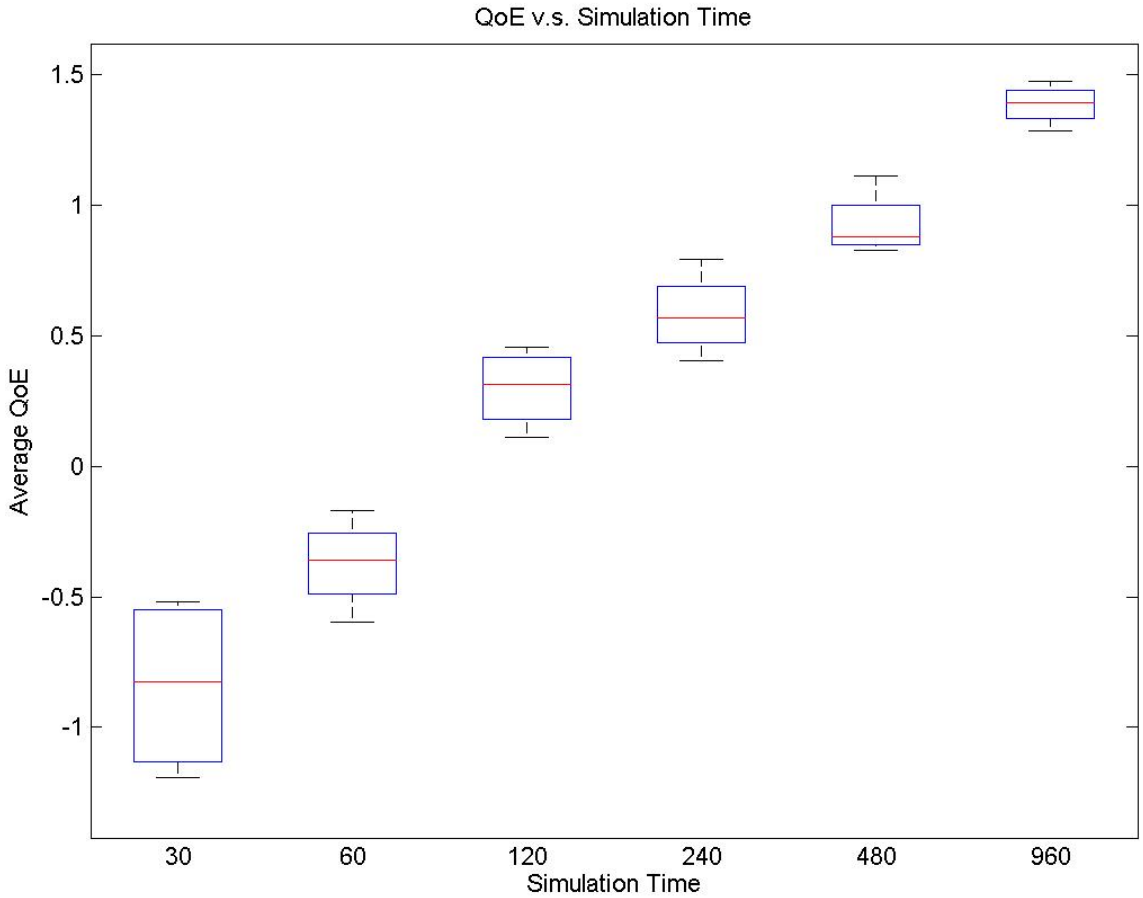


(b) contour

Figure: Conclusion, QoE and Performance

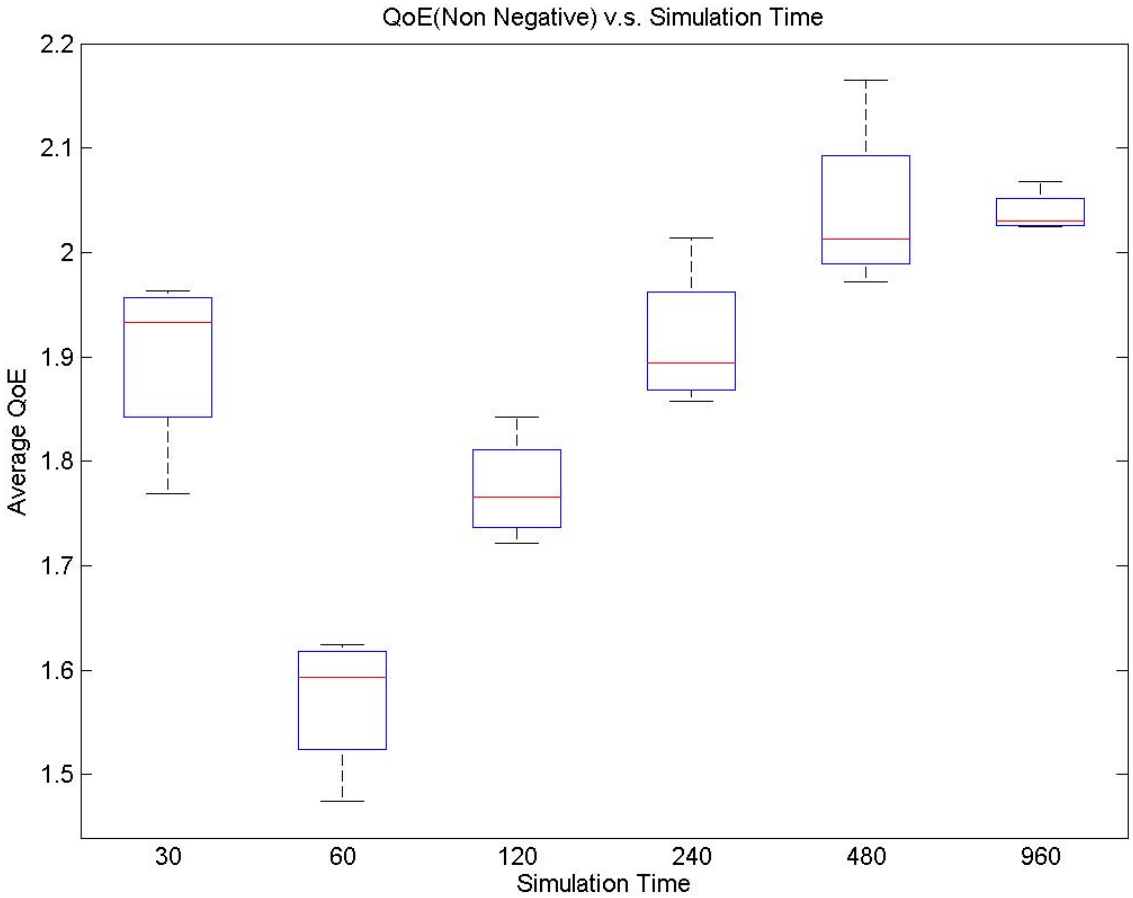
Baseline Test

QoE v.s. simulation time



Baseline Test

QoE(nonneg) v.s. simulation time



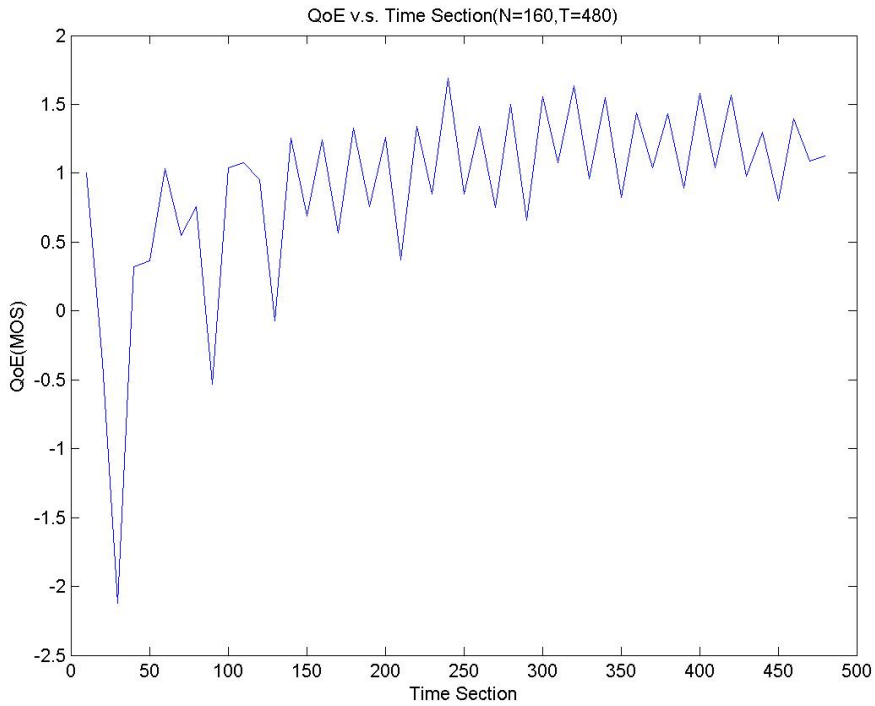
Baseline Test: A Sample Run

Configuration

- $'count\_mr' => 3749$
- $'count\_mr\_nonneg' => 2601$
- $'avg\_qoe' =>' 0.871540677419684'$
- $'avg\_qoe\_nonneg' =>' 2.01996612593628'$
- $'node\_num' =>' 160'$
- $'simu\_time' =>' 480'$
- $'core\_num' =>' 4'$
- $'task\_duration' => 8399$
- $'pdns\_duration' => 8302$

Baseline Test: A Sample Run

QoE v.s. time slot

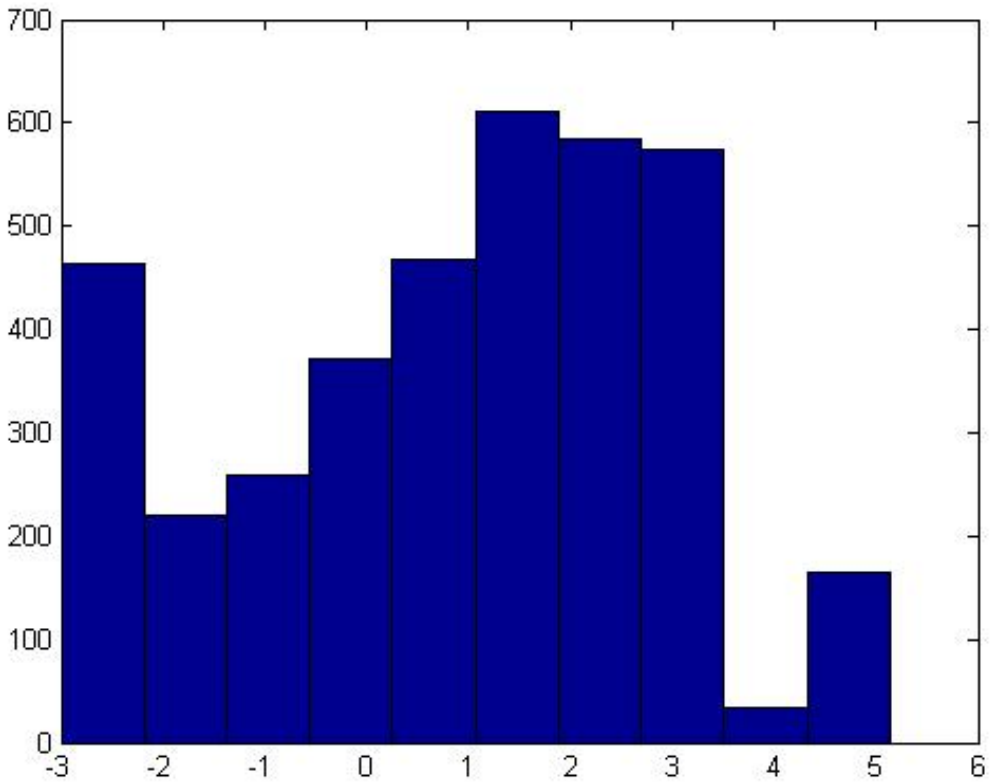


Observations:

- System bootstraping stage.
- Steady after about 250 seconds.

Baseline Test: A Sample Run

histogram of 3749 QoE reports



Version2: Unified Architecture

original framework

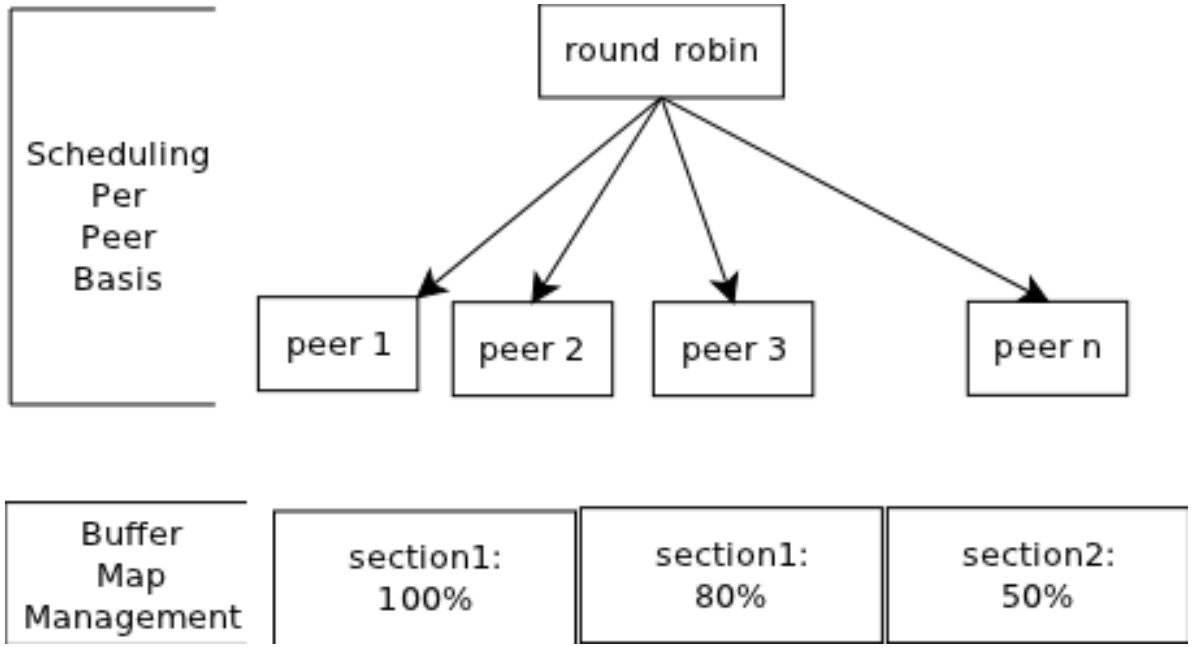


Figure: Original Architecture

Version2: Unified Architecture

unified framework

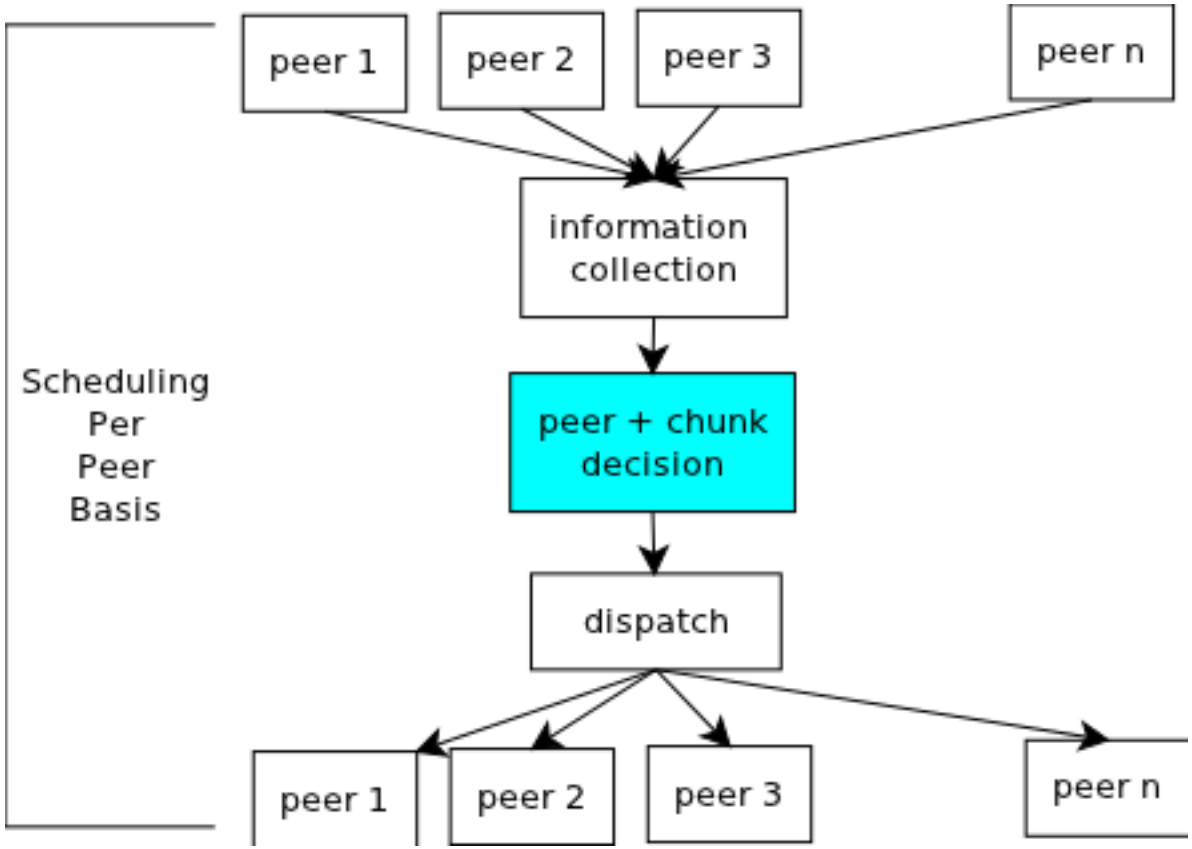


Figure: Improved Architecture

Version2: Unified Architecture

PALS like strategy

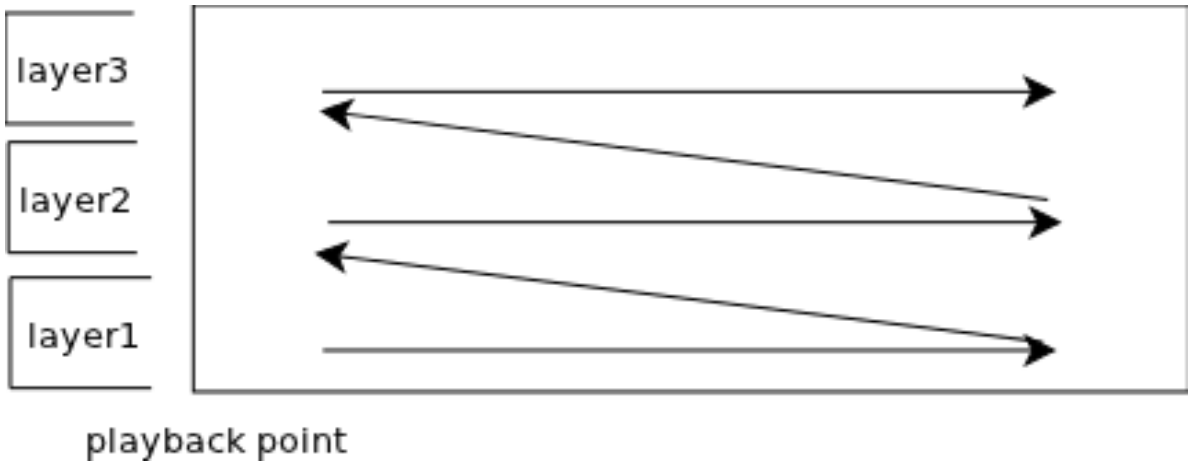


Figure: Improved Architecture

Effect:

- QoE: 0.7 $\rightarrow$ 3.3

Version3: Priority Based

framework

Table: A Sample Priority Table with Window Size = 5

	t1	t2	t3	t4	t5
layer3	11	12	13	14	15
layer2	6	7	8	9	10
layer1	1	2	3	4	5

Result:

- QoE: nearly the same.
- Performance: decrease slightly.

Version4: Scalable Window

Reason

Reason:

- From trace, many powerful peeers can get full 3 layers in the whole window.
- Give them a chance to download more, and their data far from playback pointer can server others.

Table: A Sample Priority Table with Window Size = 5+5

	t1	t2	t3	t4	t5	t6-t10
layer3	11	12	13	14	15	...
layer2	6	7	8	9	10	...
layer1	1	2	3	4	5	...

Version4: Scalable Window

Framework

Table: A Sample Priority Table with Window Size = 5+5

	t1	t2	t3	t4	t5	t6-t10
layer3	11	12	13	14	15	...
layer2	6	7	8	9	10	...
layer1	1	2	3	4	5	...

Result:

- QoE: nearly the same.
- Performance: decrease drastically.

Version5: Performance Optimization

running profile

```
====10213===profile====
Each sample counts as 0.01 seconds.
%   cumulative   self              self     total
time   seconds   seconds    calls  ms/call  ms/call  name
7.13     49.60    49.60                             TclExecuteByteCode
5.34     86.75    37.15                             Tcl_FindHashEntry
2.96    107.37    20.61                             Tcl_NewStringObj
2.57    125.23    17.86                             TclLookupSimpleVar
1.91    138.56    13.32    27165     0.49     3.75  DownloadApp::requestData3(
1.63    149.91    11.36                             HashStringKey
1.49    160.28    10.37                             ResetObjResult
1.45    170.33    10.05                             TclEvalObjvInternal
```

```
====10215===prifile====
Each sample counts as 0.01 seconds.
%   cumulative   self              self     total
time   seconds   seconds    calls  ms/call  ms/call  name
7.29     47.95    47.95                             TclExecuteByteCode
5.42     83.61    35.67                             Tcl_FindHashEntry
3.01    103.42    19.80                             Tcl_NewStringObj
2.80    121.84    18.42                             TclLookupSimpleVar
1.76    133.40    11.55    24660     0.47     3.59  DownloadApp::collect_inforam
tion_1(int, int, int, _BufferMapStatus (*) [120], std::set<unsigned int, std::le
ss<unsigned int>, std::allocator<unsigned int> >&, int&, int&)
...
0.15    544.08     1.00    27045     0.04     3.69  DownloadApp::requestData4()
...
0.02    639.08     0.12    24660     0.00     0.10  DownloadApp::check_request_a
nd_collect_inforamtion_2(_BufferMapStatus (*) [120], int, int, int&)
```

Figure: GNU Profile, Output

Version5: Performance Optimization

result

Result:

QoE: nearly the same.

Performance: 4.5h $\rightarrow$ 3.5h.

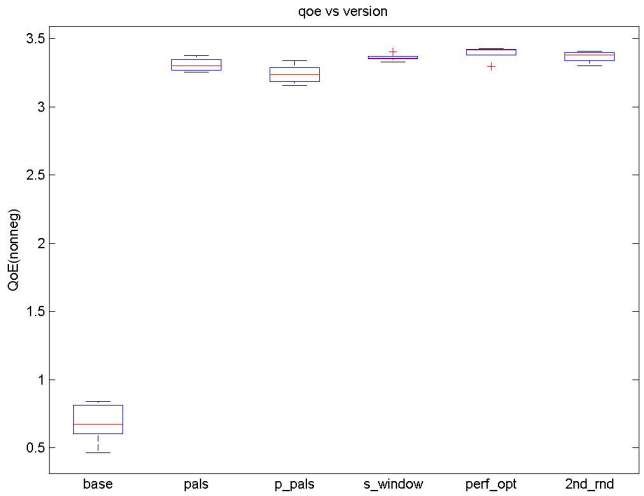
Version6: Random Second Window Section

strategy

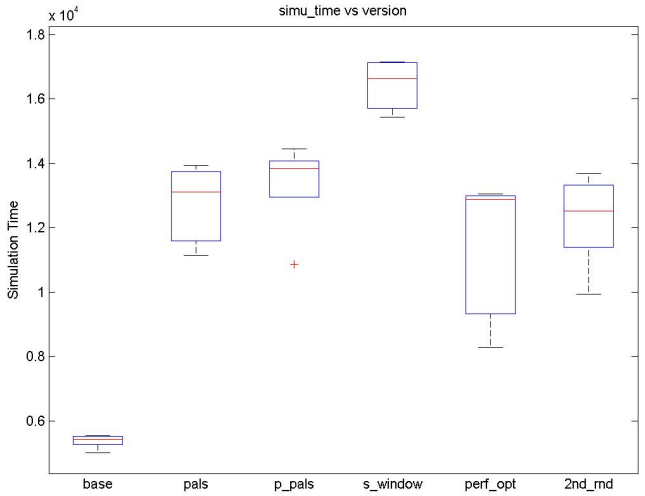
Result:

- QoE: worse.
- Performance: worse.

Conclusion
6 big versions comparison



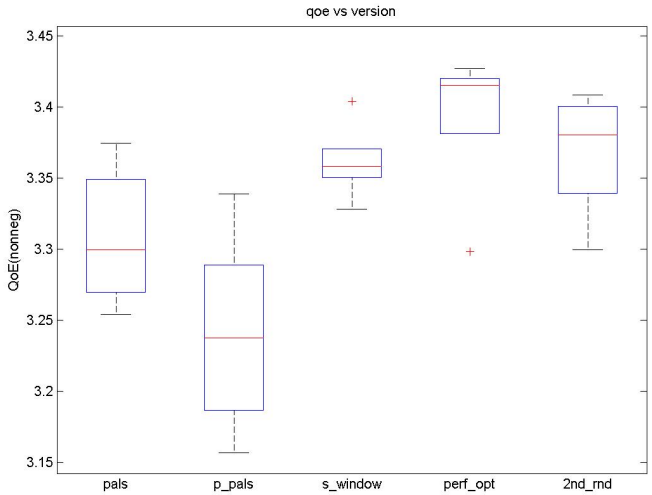
(a) qoe



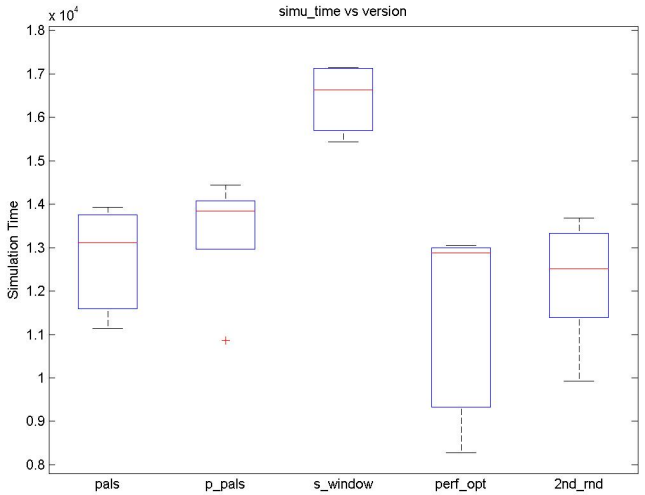
(b) time

Figure: Conclusion, QoE and Performance

Conclusion
5 enhanced versions comparision(detail)



(a) qoe



(b) time

Figure: Conclusion, QoE and Performance

Closing Remarks

- Engineering approach v.s. academic approach: where is the biggest cake?
- Time distribution:
  - 70%, literature survey.(30+ papers)
  - 15%, bugfix of the platform, environment setup.
  - 5%, first unified version(QoE:0.7→3.3, biggest improvement in this study)
  - 10%, scalable window, performance optimization, random 2nd section. (little outcome)

Toolset

find it on my homepage

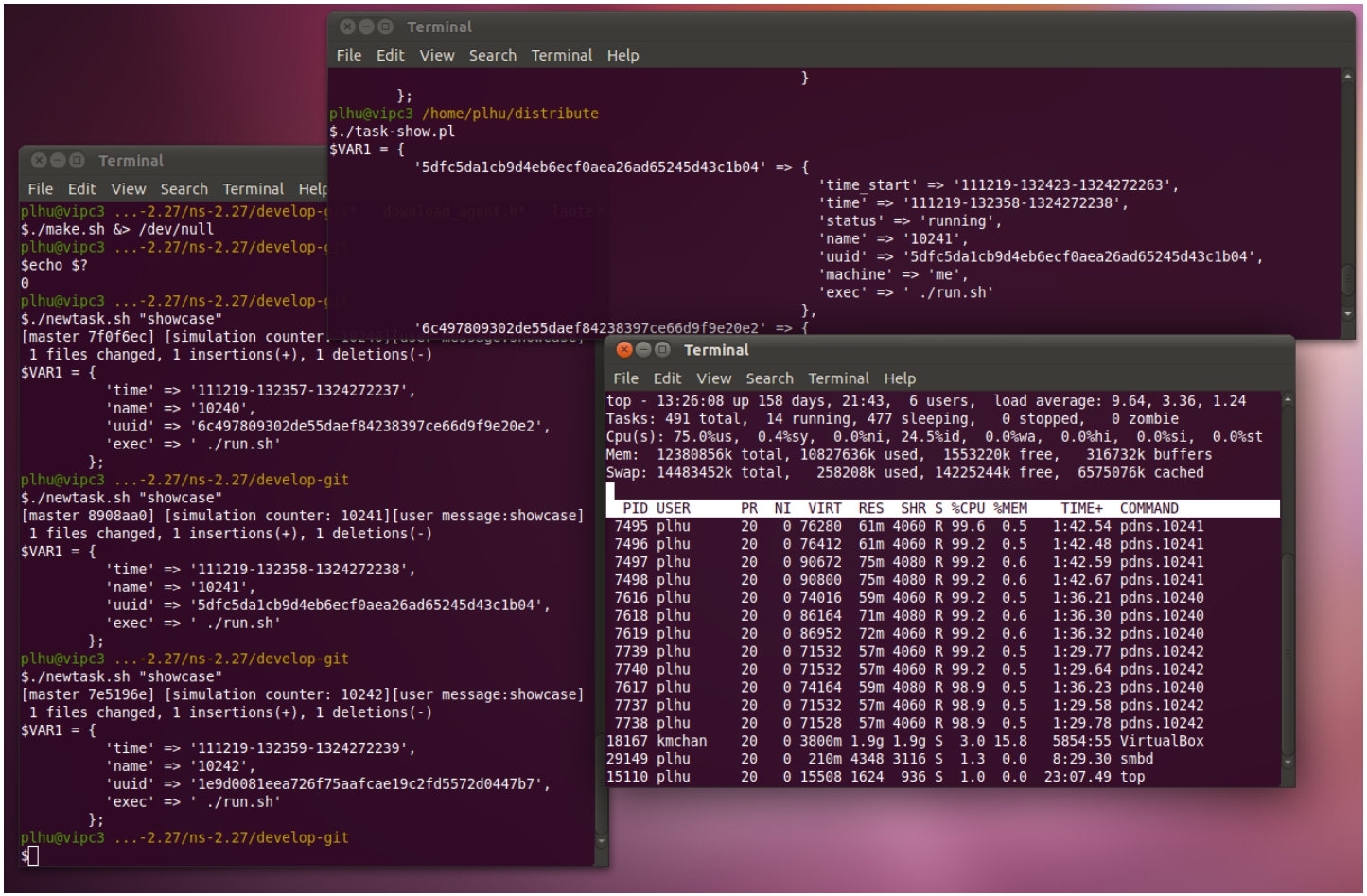


Figure: Lightweight Distributing Toolset

Thanks
some statistics

Thanks!

Some statistics:

- 6 big versions / 240 runs.
- Auxilary Scripts:
  - .sh:298 lines
  - .pl:791 lines
  - .m:133 lines
- Simulation Code Difference:
  - download_agent.cc: 1940 lines
  - download_agent.h: 301 lines
  - labtesting.tcl: 84 lines