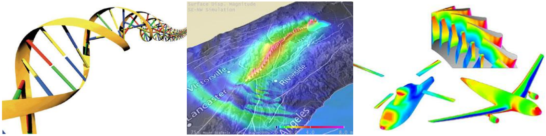
Fast and Accurate Simulations of Large-Scale Distributed Computing Systems

Pedro Velho
Équipe MESCAL

Dirigé par Jean-François Méhaut
Co-encadré par Arnaud Legrand

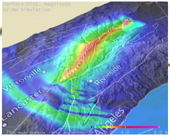
Context

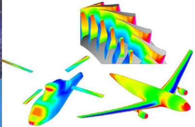
Some applications demand high computational power



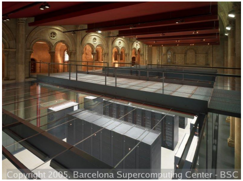
Context

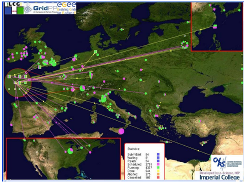
Some applications demand high computational power





Large-Scale Distributed Computing systems

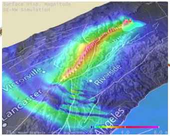


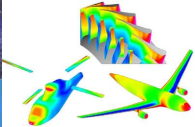


Key Features: very large scale, complex network, heterogeneity, . . .

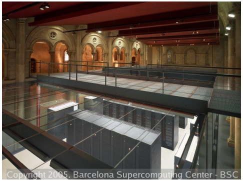
Context

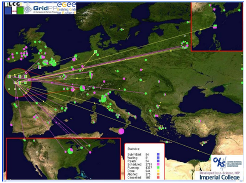
Some applications demand high computational power





Large-Scale Distributed Computing systems





Key Features: very large scale, complex network, heterogeneity, . . .

LSDC systems are complex systems that deserve scientific study

Context

LSDC as a computer science research field

- Understand the performance of LSDC systems
  (focus on time: response time, throughput, . . .)
- Need to try and compare several alternatives
- Need to reproduce the results of others to improve

Context

LSDC as a computer science research field

- Understand the performance of LSDC systems
  (focus on time: response time, throughput, ...)
- Need to try and compare several alternatives
- Need to reproduce the results of others to improve

Conducting experiments on **real** systems is hard

- Reproducibility (systems shared with others, access to the systems of others, open source, ...)
- Time consuming (tedious, requires full-fledged implementation, waste resources, ...)
- Availability (limited access to production platforms, systems may not even exist yet,...)

Context

LSDC as a computer science research field

- Understand the performance of LSDC systems
  (focus on time: response time, throughput, ...)
- Need to try and compare several alternatives
- Need to reproduce the results of others to improve

Conducting experiments on **real** systems is hard

- Reproducibility (systems shared with others, access to the systems of others, open source, ...)
- Time consuming (tedious, requires full-fledged implementation, waste resources, ...)
- Availability (limited access to production platforms, systems may not even exist yet,...)

LSDC research often resorts to simulation

Context

Simulation enables to address the previous issues

- Reproducible (deterministic execution of a sequential program)
- Complete control over the simulation process, which enables accurate comparison of alternatives
- Enables what-if analysis
- Fast results save hours (months?) of computation and labor

Context

Simulation enables to address the previous issues

- Reproducible (deterministic execution of a sequential program)
- Complete control over the simulation process, which enables accurate comparison of alternatives
- Enables what-if analysis
- Fast results save hours (months?) of computation and labor

But what is simulation ?

- Simulation = implementation of a model in a computer program
- Model = approximation of the behavior of a system

Simple models are generally fast but do they provide sound results ?

Context

Simulation enables to address the previous issues

- Reproducible (deterministic execution of a sequential program)
- Complete control over the simulation process, which enables accurate comparison of alternatives
- Enables what-if analysis
- Fast results save hours (months?) of computation and labor

But what is simulation ?

- Simulation = implementation of a model in a computer program
- Model = approximation of the behavior of a system

Simple models are generally fast but do they provide sound results ?

Simulation being an approximation, its accuracy has to be verified against real systems at hand

Context

This thesis focus on **fast** and **accurate** simulations for LSDC

- Is it possible to find a decent tradeoff between efficiency and accuracy?
- What is the validity range of simple (hence fast) models?

Context

This thesis focus on **fast** and **accurate** simulations for LSDC

- Is it possible to find a decent tradeoff between efficiency and accuracy?
- What is the validity range of simple (hence fast) models?

These questions have been addressed through the following steps

- Propose a scientific methodology that relies on systematic observation, analysis and hypothesis testing
- Apply this methodology to the conception and validation of fluid network models
- Integrate the result of this research into the open source SimGrid simulation framework

Outline

Outline

Network Simulation

Network research rely on simulation since a long time

- Standards and open-source popular projects (NS2, GTNetS, SSFNet, NS3)
  $\leadsto$ validity enforced through wide usage and reproduction by others

Network Simulation

Network research rely on simulation since a long time

- Standards and open-source popular projects (NS2, GTNetS, SSFNet, NS3)
  $\rightsquigarrow$ validity enforced through wide usage and reproduction by others
- Simulate the entire protocol stack (each packet is a simulation event)
  $\rightsquigarrow$ accuracy obtained through a very precise modeling

$\rightsquigarrow$ very **slow**, serious scalability issues for LSDC

Network Simulation

Network research rely on simulation since a long time

- Standards and open-source popular projects (NS2, GTNetS, SSFNet, NS3)
  $\rightsquigarrow$ validity enforced through wide usage and reproduction by others
- Simulate the entire protocol stack (each packet is a simulation event)
  $\rightsquigarrow$ accuracy obtained through a very precise modeling

$\rightsquigarrow$ very **slow**, serious scalability issues for LSDC

LSDC requires simplified models: two main approaches

Delay-based
Contention-based

Network Simulation

Network research rely on simulation since a long time

- Standards and open-source popular projects (NS2, GTNetS, SSFNet, NS3)
  $\rightsquigarrow$ validity enforced through wide usage and reproduction by others
- Simulate the entire protocol stack (each packet is a simulation event)
  $\rightsquigarrow$ accuracy obtained through a very precise modeling

$\rightsquigarrow$ very **slow**, serious scalability issues for LSDC

LSDC requires simplified models: two main approaches

Delay-based (P2P, LogP, ...) (fast but limited validity)

Contention-based

Network Simulation

Network research rely on simulation since a long time

- Standards and open-source popular projects (NS2, GTNetS, SSFNet, NS3)
  $\rightsquigarrow$ validity enforced through wide usage and reproduction by others
- Simulate the entire protocol stack (each packet is a simulation event)
  $\rightsquigarrow$ accuracy obtained through a very precise modeling

$\rightsquigarrow$ very **slow**, serious scalability issues for LSDC

LSDC requires simplified models: two main approaches

Delay-based (P2P, LogP, ...) (fast but limited validity)

Contention-based

- Naive packet-level (slow and inaccurate)
- Fluid (fast but accurate ?)

Network Simulation

Network research rely on simulation since a long time

- Standards and open-source popular projects (NS2, GTNetS, SSFNet, NS3)
  $\rightsquigarrow$ validity enforced through wide usage and reproduction by others
- Simulate the entire protocol stack (each packet is a simulation event)
  $\rightsquigarrow$ accuracy obtained through a very precise modeling

$\rightsquigarrow$ very **slow**, serious scalability issues for LSDC

LSDC requires simplified models: two main approaches

Delay-based (P2P, LogP, ...) (fast but limited validity)

Contention-based

- Naive packet-level (slow and inaccurate)
- Fluid (fast but accurate ?)

Fluid models are promising but their accuracy need to be assessed.

How is validation handled in the literature ?

Validation Quality in Related Work

No validation Chicsim, P2PSim, PeerSim, . . .

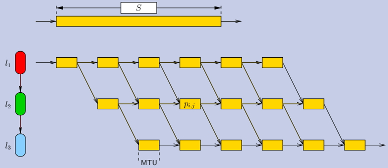
Validation Quality in Related Work

No validation Chicsim, P2PSim, PeerSim, ...

Unreliastic models or incorrect implementation

- GridSim

Naive packet level



Such naive models completely forget about the whole software and network stack.

SimGrid relied on a similar model until 2002.

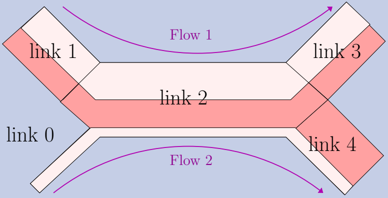
Validation Quality in Related Work

No validation Chicsim, P2PSim, PeerSim, ...

Unrealistic models or incorrect implementation

- GridSim
- OptorSim and GroudSim

Incorrect implementation




```
UNWANTED FEATURES - not really
errors but we'd rather they
weren't there
[..]
2.  All network connections use
the same fraction of bandwidth
no matter what the bottleneck
is
```

Validation Quality in Related Work

No validation Chicsim, P2PSim, PeerSim, ...

Unrealistic models or incorrect implementation

- GridSim
- OptorSim and GroudSim

"Validated"

Validation Quality in Related Work

No validation Chicsim, P2PSim, PeerSim, ...

Unrealistic models or incorrect implementation

- GridSim
- OptorSim and GroudSim

"Validated"

- Shallow description, unavailable or broken code:

Common practice in the field

Most people build their own "ad-hoc" solutions. *Naicken, Stephen et Al., Towards Yet Another Peer-to-Peer Simulator, HET-NETs'06*:

From 141 P2P sim.papers, 30% use a custom tool, 50% don't report used tool.

Validation Quality in Related Work

No validation Chicsim, P2PSim, PeerSim, ...

Unrealistic models or incorrect implementation

- GridSim
- OptorSim and GroudSim

"Validated"

- Shallow description, unavailable or broken code:
  - Dimemas (1992 project open-sourced very recently)
  - PMAC convolver, PSins (last version revealed to work only with specific version of MPI), ...

"Magical" formulas and parameters

"Convolutions can be arbitrarily complex depending upon how many features of the application and the machine are being accounted for."

When the code is not available, it is not even possible to check what the authors had in mind.

Validation Quality in Related Work

No validation Chicsim, P2PSim, PeerSim, ...

Unrealistic models or incorrect implementation

- GridSim
- OptorSim and GroudSim

"Validated"

- Shallow description, unavailable or broken code:
  - Dimemas (1992 project open-sourced very recently)
  - PMAC convolver, PSins (last version revealed to work only with specific version of MPI), ...
- Optimistic validation LogPSim, Bigsim

Avoid difficult situations

"*Our simulator ignores congestion in the network and assumes full effective bisection bandwidth.*"

Although more evolved models (modeling contention) have been implemented, there is no publication on it yet and according to the author, it does not seem to improve much accuracy.

Validation Quality in Related Work

No validation Chicsim, P2PSim, PeerSim, ...

Unrealistic models or incorrect implementation

- GridSim
- OptorSim and GroudSim

"Validated"

- Shallow description, unavailable or broken code:
  - Dimemas (1992 project open-sourced very recently)
  - PMAC convolver, PSins (last version revealed to work only with specific version of MPI), ...
- Optimistic validation
  LogPSim, Bigsim

Avoid difficult situations

Resolution	*Elements*	*Storage*	*1 Processor*	*1K Processors*	*100K Processors*	*1M Processors*
1cm	10^6	40MB	1s	1ms	10us	1us
1mm	10^9	40GB	1000s	1s	10ms	1ms

Table 1. Meshes for a 1m^3 domain, storage requirements and time per timestep on various machines.

their time communicating and efficiency will be very low.

Communication latency can be hidden to a large extent with the technique of "**processor virtualization**", in which the problem is decomposed into more pieces than processors, and the pieces scheduled dynamically based on which messages are available. CHARM++ and the FEM framework fully support virtualization, and in fact require no extra user code for a virtualized run.

Another complementary approach to handle communication latency is the ghost cell expansion method [12], where redundant computations around each processor's border are used to decrease the frequency of message exchange. This multiple-ghost approach has only been implemented for structured grids, however, and the extension to unstructured grids, while conceptually straightforward, would be complicated to implement.

4.1.3 Bottlenecks on Large Machines

In summary, there are a number of practical bottlenecks to execution on very large machines. First, large meshes must be generated; this is difficult with today's tools. Second, the meshes must be partitioned for parallel execution. Finally, the resulting computation may still have small grainsize, so messaging performance is important.

Our runs with BigSim also exposed a number of unexpected bottlenecks and limitations to scalability. For example, the serial partitioning library we use consumes memory proportional to the number of output pieces, not the total size of the mesh; so even our 4GB machine ran out of memory when partitioning a relatively small 5M element mesh into more than 16K pieces. Hopefully ParMetis will solve this problem.

Similarly, even though our MPI implementation, AMPI, was designed to be scalable, while trying to simulate very large machines we discovered our implementation used P^2 total memory for P processors. The culprit was a simple linear message ordering table kept by each processor, because the table's length was proportional to the number of processors. For today's machines, where P=1000, the total amount of memory used was 16MB; but for P=100,000, the tables would use 160GB! The solution was to break the tables into (software) pages and only allocate pages when referenced; this dramatically reduces the storage requirements for large machines because most processors only communicate directly with a small subset of other processors.

5 Performance

We first present results of validation tests using BigNetSim on Lemieux [13] at Pittsburgh Supercomputer Center. We then present results of performance prediction and performance analysis of some real world FEM applications using the simulator. Finally we will present the scaling performance of the BigNetSim simulator itself.

5.1 Validation

We have compared the actual running time of a simple 7-point stencil computation with a 3-D decomposition written in MPI with our simulation of it using BigNetSim. In the program, every chunk of data communicates with its six neighbors in three dimensions. The Jacobi relaxation computation is performed, and the maximum error is calculated via **MPI_Allreduce**.

The result is shown in Table 2 for a problem with fixed size in all runs. The first row shows the running time of the MPI program on 32 to 256 processors; the second row shows the predicted running time using BigNetSim offline on a Linux cluster. The network parameters are based on Quadrics network specifications. It shows that the simulated execution time is close to the actual execution time.

Processors	32	64	128	256
Actual run time (s)	2.21	1.07	0.48	0.26
Predicted time (s)	2.35	1.16	0.55	0.30

Table 2. Actual vs. predicted time

5.2 FEM

We studied the performance of a CHARM++ FEM Framework program, which performs a simple 2D structural simulation on an unstructured triangle mesh. We chose a relatively small problem with a 5 million element mesh, so as to stress efficiency issues. Because our 2D elements take a little under a microsecond of CPU time per timestep, this is less than 5 seconds of serial work per timestep.

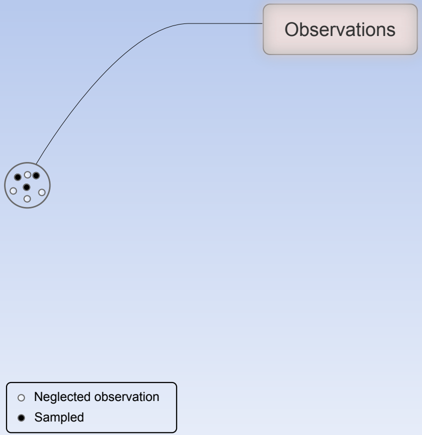
Figure 5 shows the predicted execution time per step, simulating 125 to 16,000 processors using only 32 Lemieux processors. The time per step is 23.5 milliseconds for 125 processors and drops to 640 microseconds on 16,000 processors. Figure 6 is the corresponding speedup, normalized based on the 125 processor time. It shows that the program can scale well to at least several thousands of processors.

Beyond several thousand processors, when the simulated time per step drops below a few milliseconds, the parallel efficiency begins to drop. Sub-millisecond cycle times are indeed extremely challenging even on today's small machines, and we continue to seek methods to improve this performance on even larger machines.

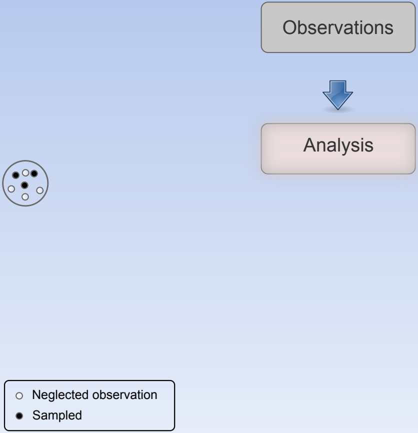
We also demonstrate the benefits of processor virtualization in CHARM++ for the same FEM program. We use different numbers of MPI virtual processors, each with a separate chunk of the problem mesh, on each simulated processor. Virtualization allows dynamic overlap of computa-

Methodology

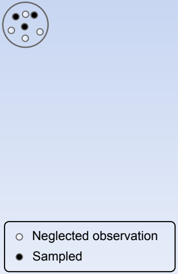
Methodology

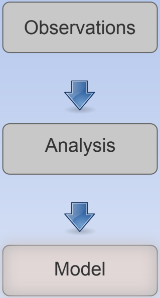


Methodology

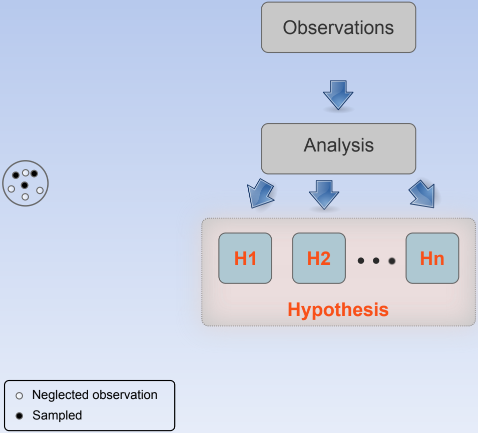


Methodology

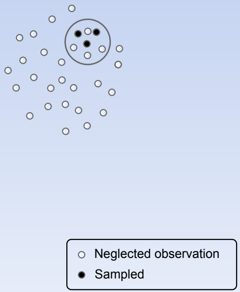


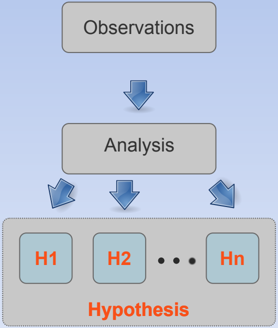


Methodology

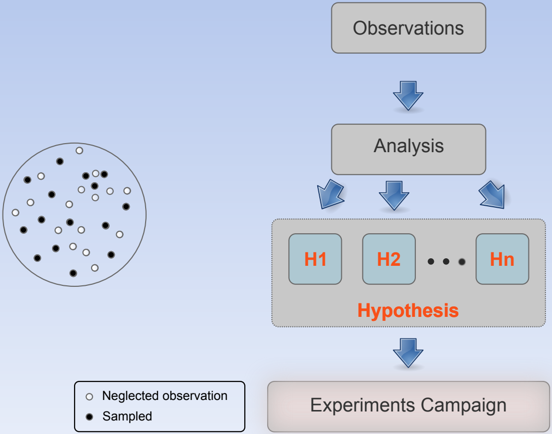


Methodology

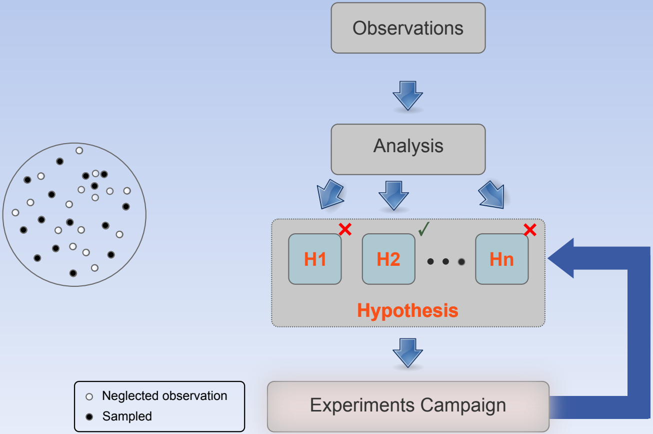




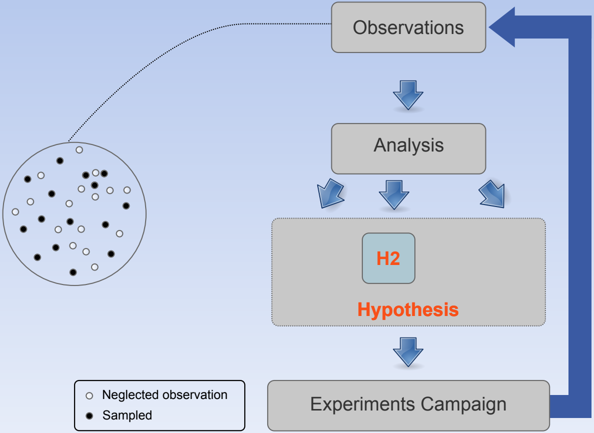
Methodology



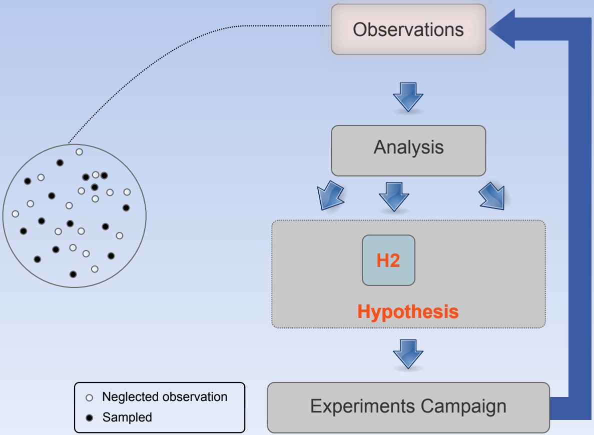
Methodology



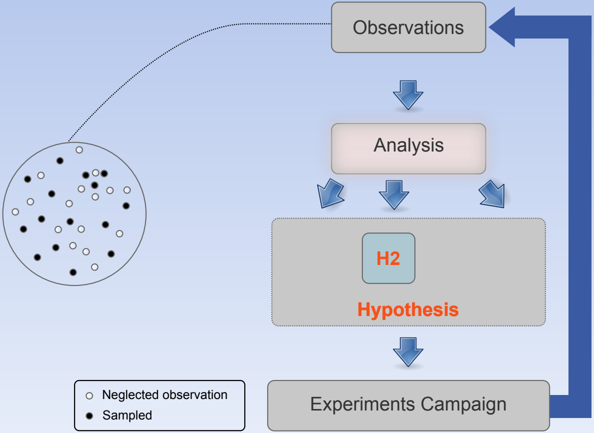
Methodology



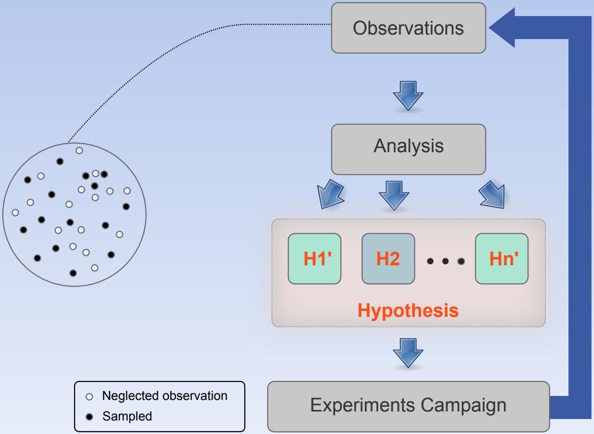
Methodology



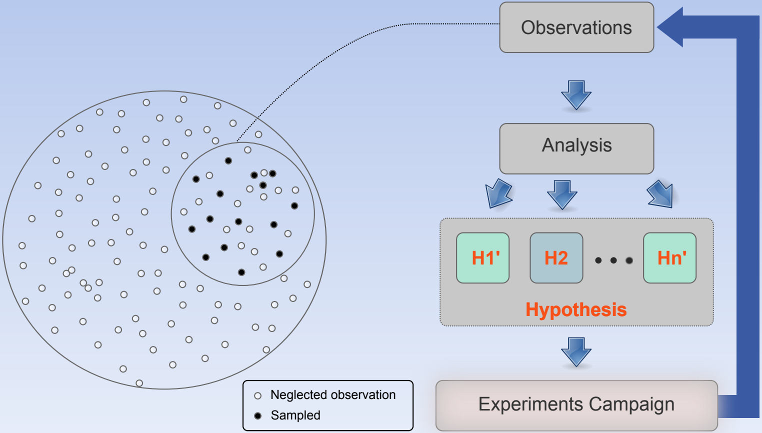
Methodology



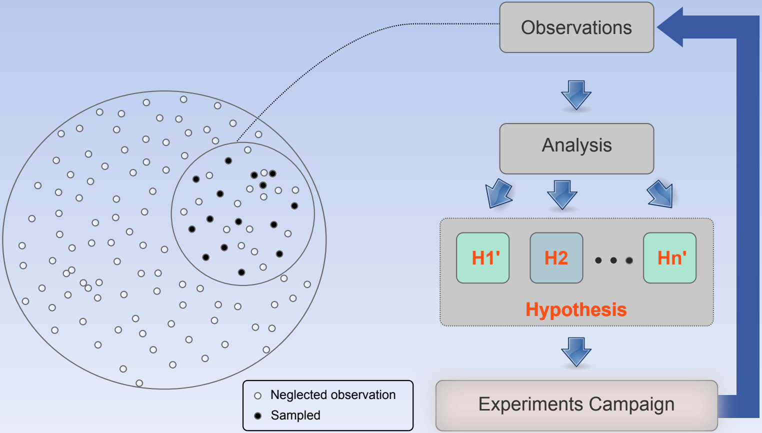
Methodology



Methodology

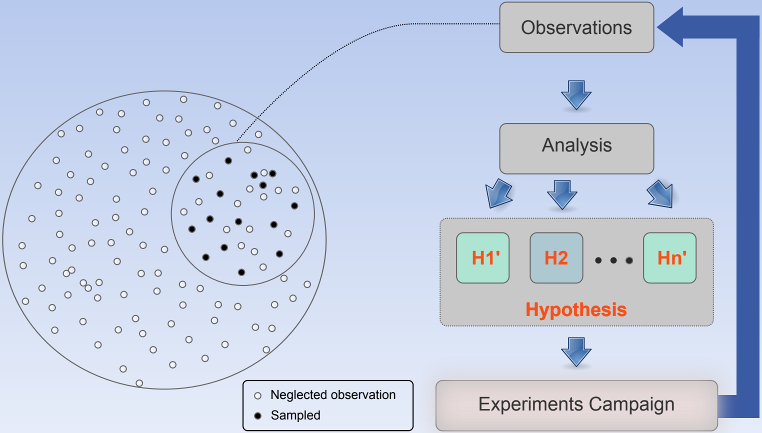


Methodology



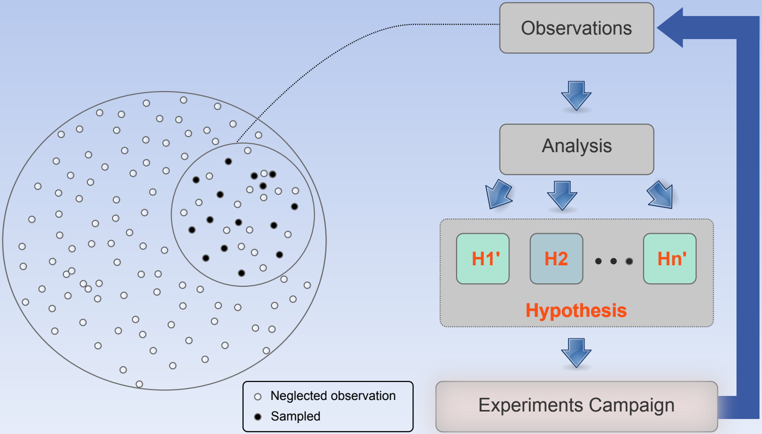
1. Validation is a cyclic process

Methodology



1. Validation is a cyclic process
2. Experiments should be designed to objectively prove or disprove an hypothesis

Methodology



1. Validation is a cyclic process
2. Experiments should be designed to objectively prove or disprove an hypothesis
3. Rejected hypothesis provide generally much more insight than accepted ones

How to perform validation experiments ?

We need to compare the outcome of our simple models for a wide variety of configurations.

How can we compare to such configurations ?

Real system **hard to instantiate** + time consuming, reproducibility, . . .

How to perform validation experiments ?

We need to compare the outcome of our simple models for a wide variety of configurations.

How can we compare to such configurations ?

~~Real system~~ ~~hard to instantiate + time consuming, reproducibility, ...~~

Emulation just **as biased as an invalidated simulation** (especially in such a context)

How to perform validation experiments ?

We need to compare the outcome of our simple models for a wide variety of configurations.

How can we compare to such configurations ?

~~Real system~~ ~~hard to instantiate + time consuming, reproducibility, ...~~

~~Emulation~~ ~~just as biased as an invalidated simulation (especially in such a context)~~

Packet-level simulator tested for year in the network community and highly configurable

How to perform validation experiments ?

We need to compare the outcome of our simple models for a wide variety of configurations.

How can we compare to such configurations ?

~~Real system~~ ~~hard to instantiate + time consuming, reproducibility, ...~~

~~Emulation~~ ~~just as biased as an invalidated simulation (especially in such a context)~~

Packet-level simulator tested for year in the network community and highly configurable

For those reasons we used GTNetS (a network simulator, that had been integrated to SimGrid for this purpose) as a comparison point.

In the following, we try to devise a good model for predicting communication times on an heterogeneous network using TCP Reno.

Outline

Basics of fluid modeling

Basic model for a single link and a single message
(physical bandwidth $= B$, physical latency $= L$, size $= S$)

$$T = \frac{S}{B} + L$$

Such naive model ignores the protocol overhead and peculiarities

Basics of fluid modeling

Basic model for a single link and a single message
(physical bandwidth $= B$, physical latency $= L$, size $= S$)

$$T = \frac{S}{B} + L$$

Such naive model ignores the protocol overhead and peculiarities

Bandwidth sharing Share bandwidth every time a new flow appears or disappears

Basics of fluid modeling

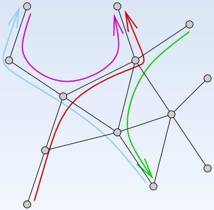
Basic model for a single link and a single message
(physical bandwidth $= B$, physical latency $= L$, size $= S$)

$$T = \frac{S}{B} + L$$

Such naive model ignores the protocol overhead and peculiarities

Bandwidth sharing Share bandwidth every time a new flow appears or disappears

- Setting a set of flows $\mathcal{F}$ and a set of links $\mathcal{L}$
- Constraints For all link j: $\displaystyle\sum_{\textbf{if flow i uses link j}} \rho_i \leqslant C_j$



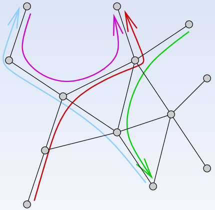
Basic model for a single link and a single message (physical bandwidth $= B$, physical latency $= L$, size $= S$)

$$T = \frac{S}{B} + L$$

Such naive model ignores the protocol overhead and peculiarities

Bandwidth sharing Share bandwidth every time a new flow appears or disappears

- Setting a set of flows $\mathcal{F}$ and a set of links $\mathcal{L}$
- Constraints For all link j: $\displaystyle\sum_{\textbf{if flow i uses link j}} \rho_i \leqslant C_j$
- Objective function
  - Max-Min $\max(\min(\rho_i))$



Basics of fluid modeling

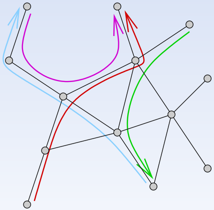
Basic model for a single link and a single message
(physical bandwidth $= B$, physical latency $= L$, size $= S$)

$$T = \frac{S}{B} + L$$

Such naive model ignores the protocol overhead and peculiarities

Bandwidth sharing Share bandwidth every time a new flow appears or disappears

- Setting a set of flows $\mathcal{F}$ and a set of links $\mathcal{L}$
- Constraints For all link j: $\displaystyle\sum_{\textbf{if flow i uses link j}} \rho_i \leqslant C_j$
- Objective function
  - Max-Min $\max(\min(\rho_i))$ (TCP?)
  - Using a "*microscopic*" analysis,
    Low proved Reno $\sim \max(\sum(\log(\rho_i)))$



Basics of fluid modeling

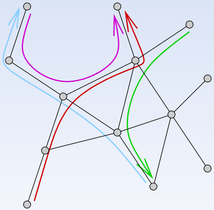
Basic model for a single link and a single message
(physical bandwidth $= B$, physical latency $= L$, size $= S$)

$$T = \frac{S}{B} + L$$

Such naive model ignores the protocol overhead and peculiarities

Bandwidth sharing Share bandwidth every time a new flow appears or disappears

- Setting a set of flows $\mathcal{F}$ and a set of links $\mathcal{L}$
- Constraints For all link j: $\sum_{\text{if flow i uses link j}} \rho_i \leqslant C_j$
- Objective function
  - Max-Min $\max(\min(\rho_i))$ (TCP?)
  - Using a "*microscopic*" analysis,
    Low proved Vegas $\sim \max(\sum(\arctan(\rho_i)))$



Single link, single flow

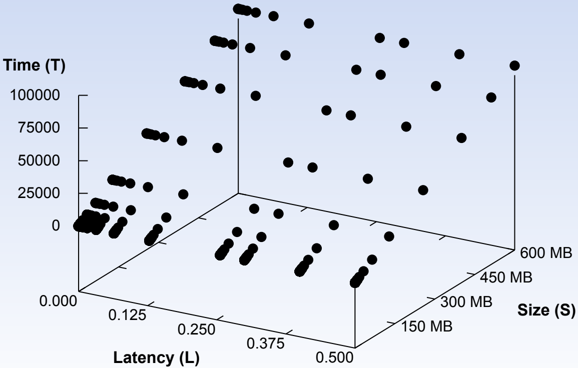
Hypothesis: for large messages, the time is linear with respect to message size

- Bandwidth fixed to low value, 56Kbps, size and latency varied
- Measure the time obtained with GTNetS

Single link, single flow

Hypothesis: for large messages, the time is linear with respect to message size

- Bandwidth fixed to low value, 56Kbps, size and latency varied
- Measure the time obtained with GTNetS



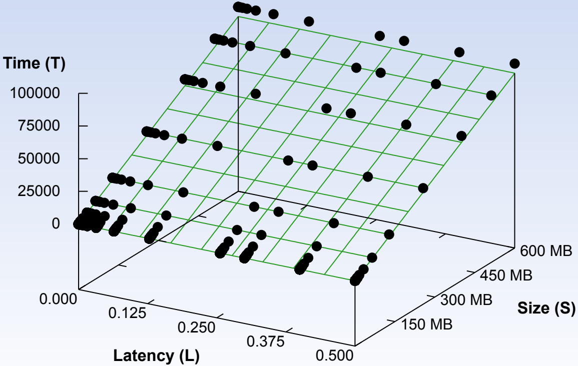
Single link, single flow

Hypothesis: for large messages, the time is linear with respect to message size

- Bandwidth fixed to low value, 56Kbps, size and latency varied
- Measure the time obtained with GTNetS

Naive model $T = \frac{S}{B} + L$ ✓

T = Time, S = Size, B = Bandwidth, L = Latency



Single link, single flow

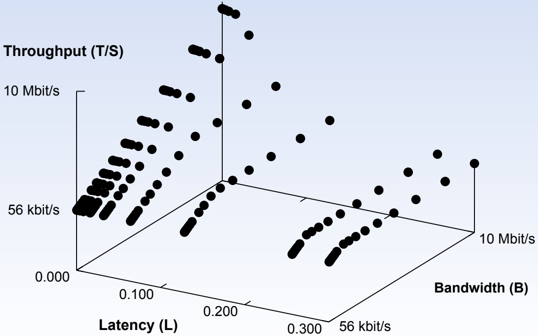
Hypothesis: effective bandwidth depends only on link physical bandwidth

- $Throughput = S/T$ is a good approximation of effective bandwidth when S is large
- Fixed message size to 10 MB, bandwidth and latency varied
- Measure the time obtained with GTNetS

Single link, single flow

Hypothesis: ~~effective bandwidth depends only on link physical bandwidth~~

- $Throughput = S/T$ is a good approximation of effective bandwidth when S is large
- Fixed message size to 10 MB, bandwidth and latency varied
- Measure the time obtained with GTNetS

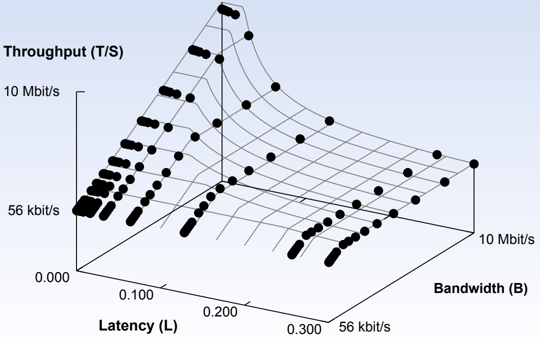


Hypothesis: ~~effective bandwidth depends only on link physical bandwidth~~

- $Throughput = S/T$ is a good approximation of effective bandwidth when S is large
- Fixed message size to 10 MB, bandwidth and latency varied
- Measure the time obtained with GTNetS

Less Naive model $T = \frac{S}{\min(B, \frac{W}{RTT})} + L$ ✓

W = Maximum TCP Window, RTT = Round Trip Time



Single link, single flow

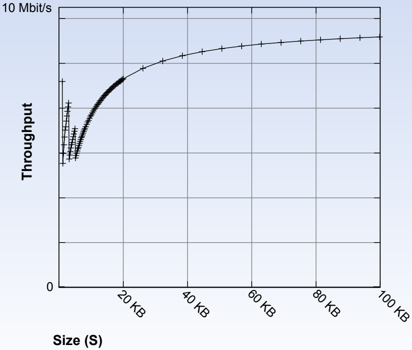
Hypothesis: effective bandwidth does not depend on message size

Naive model $\dfrac{S}{T} = \dfrac{S}{\frac{S}{B} + L}$

Single link, single flow

Hypothesis: effective bandwidth does not depend on message size

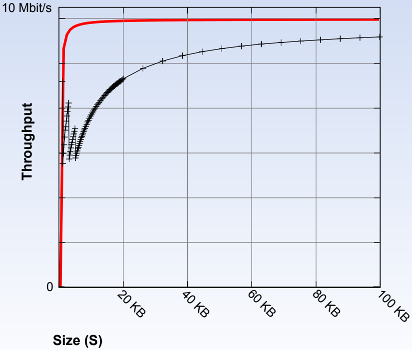
Naive model $\dfrac{S}{T} = \dfrac{S}{\frac{S}{B} + L}$



Single link, single flow

Hypothesis: ~~**effective bandwidth does not depend on message size**~~

Naive model $\frac{S}{T} = \frac{S}{\frac{S}{B} + L}$ Model does not hold.

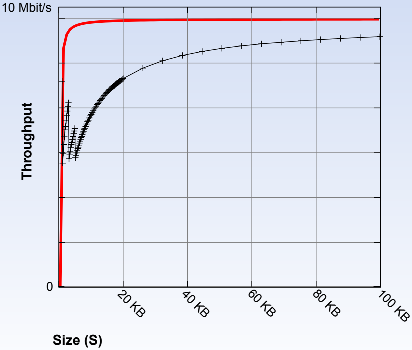


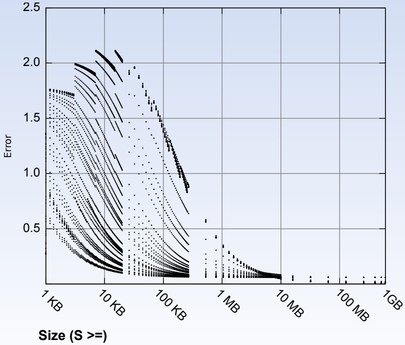
Single link, single flow

Hypothesis: ~~effective bandwidth does not depend on message size~~

Naive model $\frac{S}{T} = \frac{S}{\frac{S}{B} + L}$

Model does not hold.
Especially for small messages



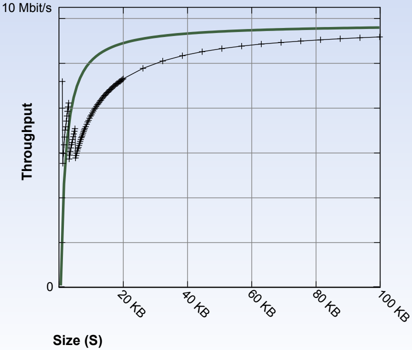


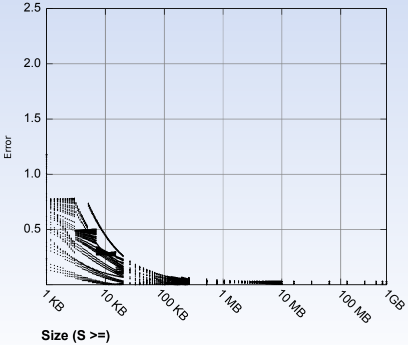
Single link, single flow

Hypothesis: ~~**effective bandwidth does not depend on message size**~~

Naive model $\frac{S}{T} = \frac{S}{\frac{S}{B} + L}$ Model does not hold.
Especially for small messages

- A linear approximation still leads to good results
- $T = \frac{S}{\min(\beta.B, \frac{W}{2.RTT})} + \alpha.L$ ✓





Dumbbell topology, two flows

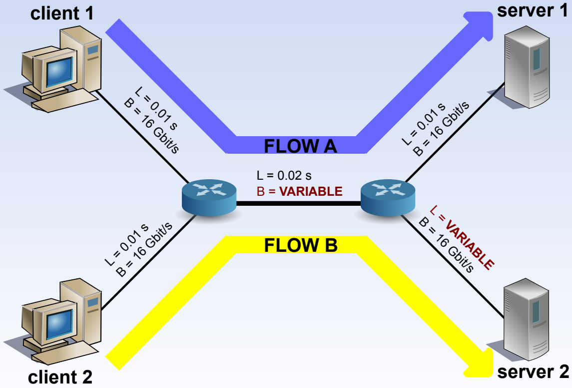
Hypothesis: Bottleneck links are proportionally shared with respect to flow RTT

Dumbbell topology, two flows

Hypothesis: Bottleneck links are proportionally shared with respect to flow RTT

$$RTT_A.\rho_A = RTT_B.\rho_B \textbf{ where } RTT_i \approx \sum_{\text{flow i uses link j}} (L_j) \text{ (naive model)}$$

- Longer flows (higher latency) will receive slightly less bandwidth

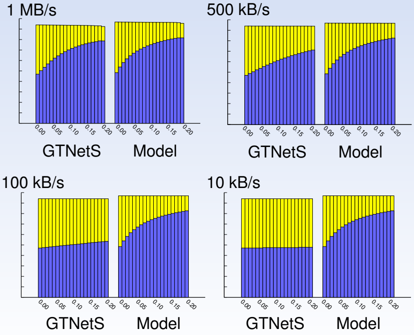


Dumbbell topology, two flows

Hypothesis: Bottleneck links are proportionally shared with respect to flow RTT

$$RTT_A.\rho_A = RTT_B.\rho_B \textbf{ where } RTT_i \approx \sum_{\text{flow i uses link j}} (L_j) \text{ (naive model)}$$

- Longer flows (higher latency) will receive slightly less bandwidth

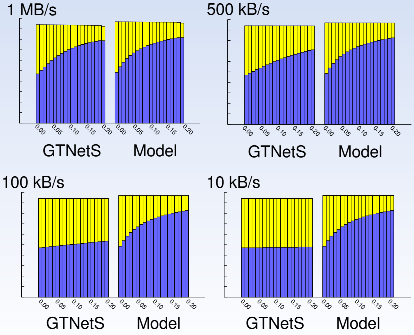


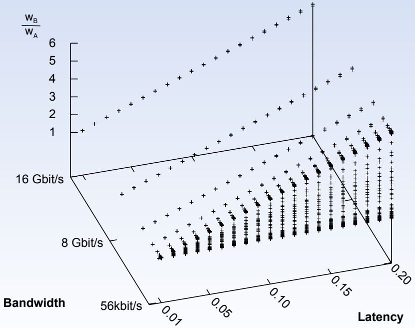
Dumbbell topology, two flows

Hypothesis: Bottleneck links are proportionally shared with respect to flow RTT

$$RTT_A.\rho_A = RTT_B.\rho_B \textbf{ where } \cancel{RTT_i \approx \sum_{\text{flow i uses link j}} (L_j)} \text{ (naive model)}$$

- Longer flows (higher latency) will receive slightly less bandwidth
- However, bandwidth also matters



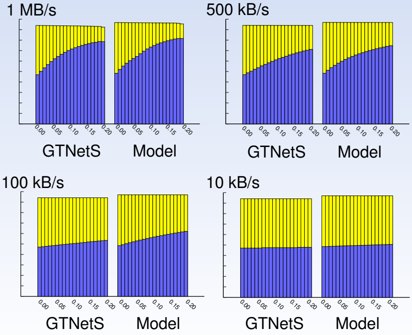


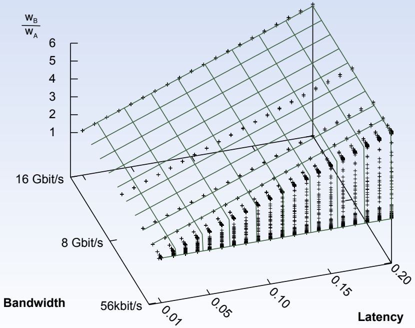
Dumbbell topology, two flows

Hypothesis: Bottleneck links are proportionally shared with respect to flow RTT

$$RTT_A.\rho_A = RTT_B.\rho_B \textbf{ where } \cancel{RTT_i \approx \sum_{\text{flow i uses link j}} (L_j)} \text{ (naive model)}$$

- Longer flows (higher latency) will receive slightly less bandwidth
- However, bandwidth also matters
- Simple fix: $RTT_i \approx \sum_{\text{flow i uses link j}} \left(\frac{M}{B_j} + L_j\right)$ ✓





Random

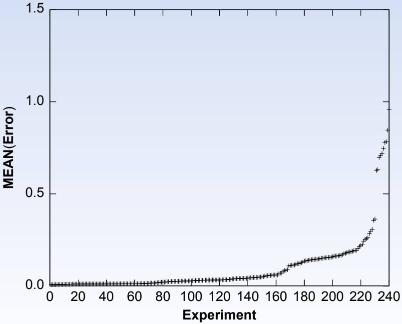
Hypothesis: our new model is valid for a wide range of settings

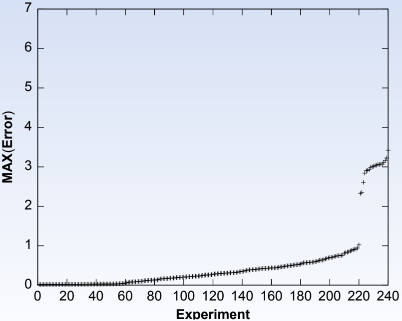
- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

Random

Hypothesis: ~~**our new model is valid for a wide range of settings**~~

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

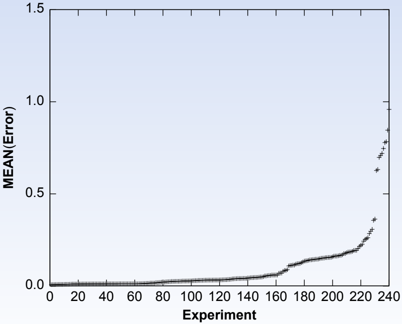


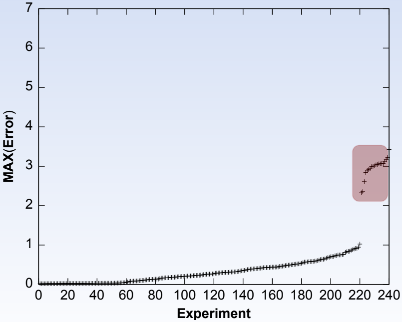


Random

Hypothesis: ~~our new model is valid for a wide range of settings~~

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform



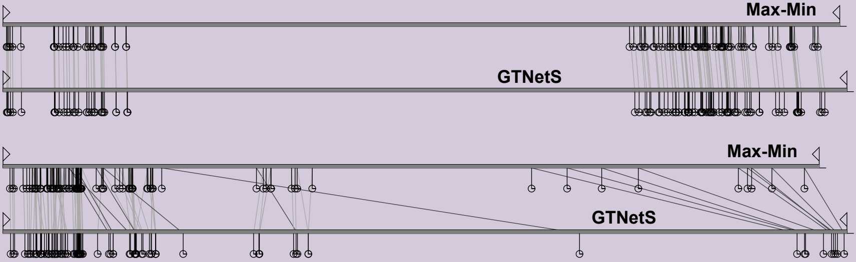


Random

Hypothesis: ~~**our new model is valid for a wide range of settings**~~

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

Comparing completion times instead of bandwidth share

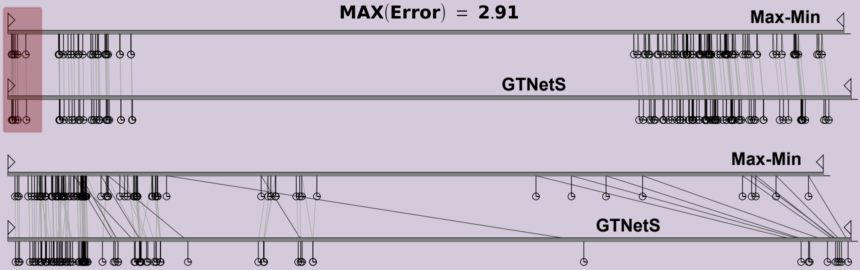


Random

Hypothesis: ~~**our new model is valid for a wide range of settings**~~

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

Comparing completion times instead of bandwidth share

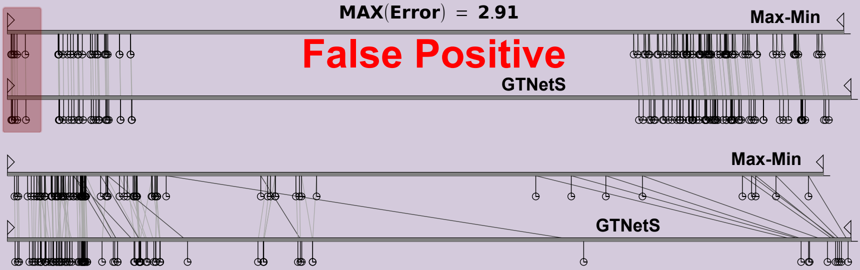


Random

Hypothesis: ~~**our new model is valid for a wide range of settings**~~

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

Comparing completion times instead of bandwidth share

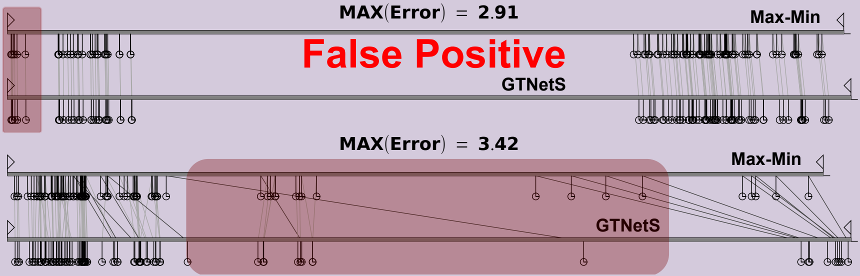


Random

Hypothesis: ~~our new model is valid for a wide range of settings~~

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

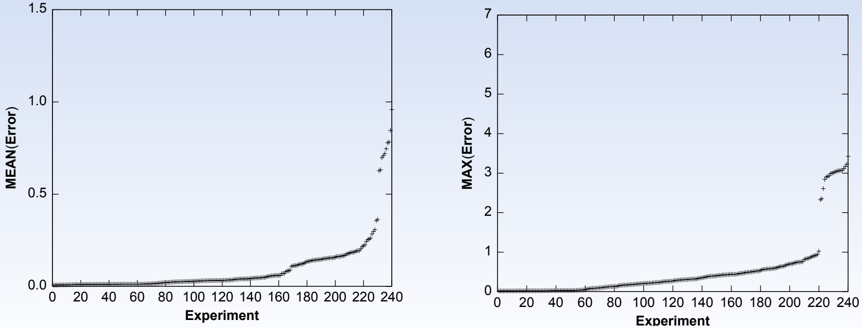
Comparing completion times instead of bandwidth share



Random

Hypothesis: ~~**our new model is valid for a wide range of settings**~~

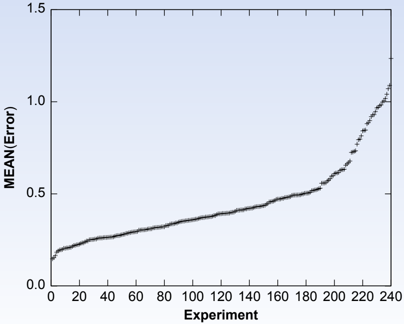
- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

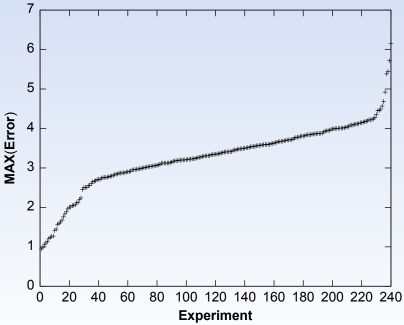


Random

Hypothesis: ~~our new model is valid for a wide range of settings~~

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform
  - **Ensure that we have contention and that flows are not limited by latency**

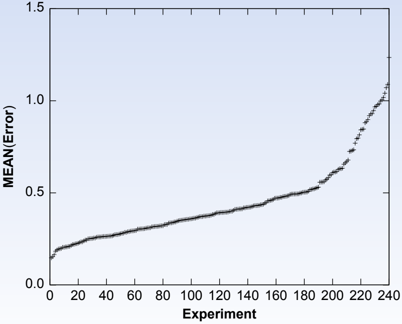


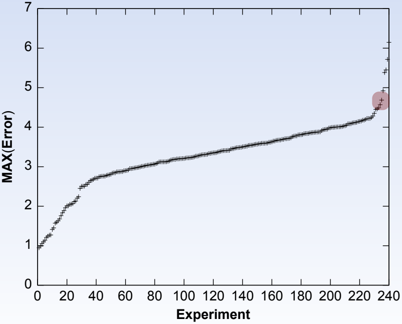


Random

Hypothesis: ~~our new model is valid for a wide range of settings~~

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform
  - **Ensure that we have contention and that flows are not limited by latency**

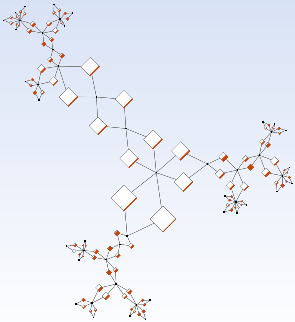




Random

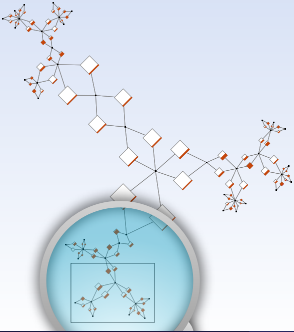
Hypothesis: ~~our new model is valid for a wide range of settings~~

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform
  - **Ensure that we have contention and that flows are not limited by latency**



Hypothesis: ~~our new model is valid for a wide range of settings~~

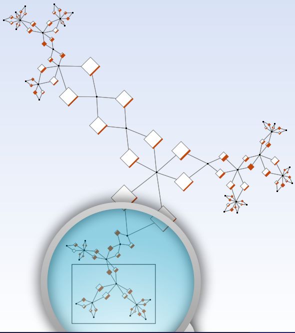
- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform
  - **Ensure that we have contention and that flows are not limited by latency**

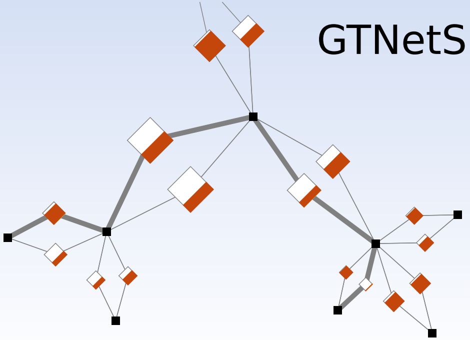


Random

Hypothesis: ~~**our new model is valid for a wide range of settings**~~

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform
  - **Ensure that we have contention and that flows are not limited by latency**

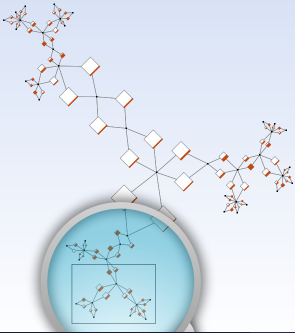


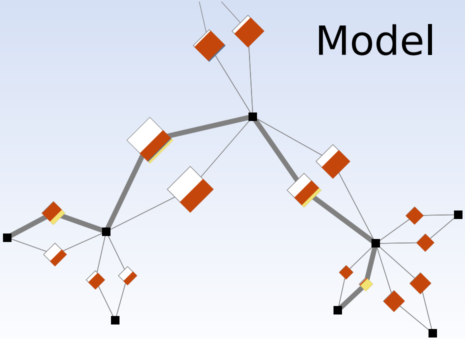


Random

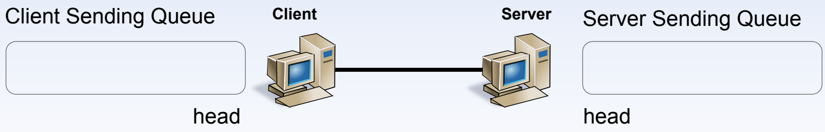
Hypothesis: ~~our new model is valid for a wide range of settings~~

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform
  - **Ensure that we have contention and that flows are not limited by latency**



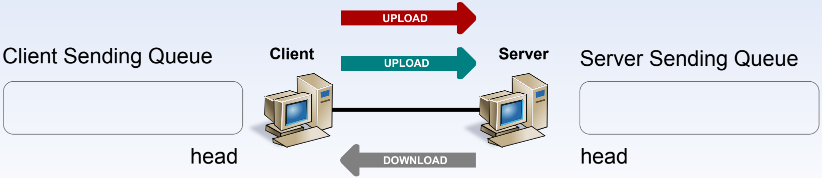


Single link, cross-traffic



Single link, cross-traffic

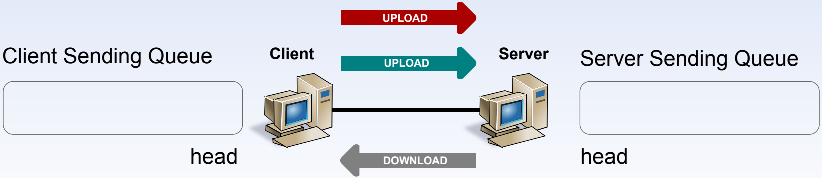
What is the bandwidth share of such a situation ?



Single link, cross-traffic

What is the bandwidth share of such a situation ?

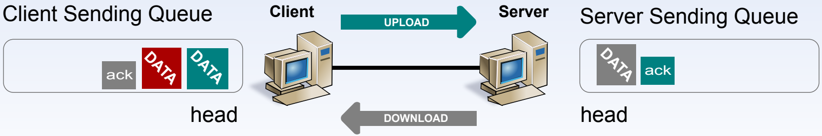
Every flow gets B/2 !!!



Single link, cross-traffic

What is the bandwidth share of such a situation ?

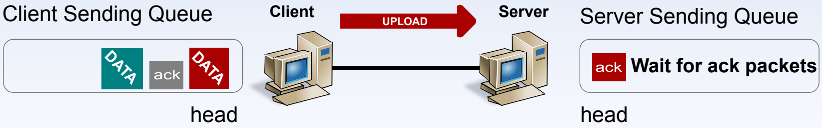
- Ack packets get compressed by data packets (which are bigger)



Single link, cross-traffic

What is the bandwidth share of such a situation ?

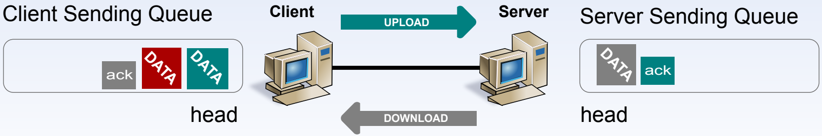
- Ack packets get compressed by data packets (which are bigger)
- The download can transmit only half of the time



Single link, cross-traffic

What is the bandwidth share of such a situation ?

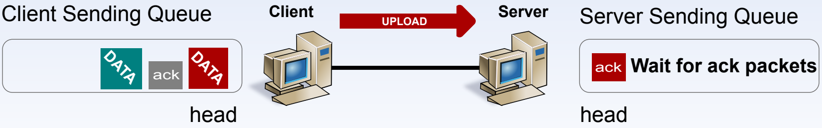
- Ack packets get compressed by data packets (which are bigger)
- The download can transmit only half of the time
- Resulting bandwidth of each flow $\approx \frac{C}{2}$



Single link, cross-traffic

What is the bandwidth share of such a situation ?

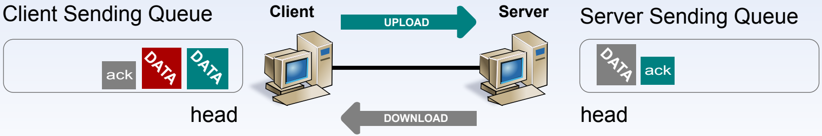
- Ack packets get compressed by data packets (which are bigger)
- The download can transmit only half of the time
- Resulting bandwidth of each flow $\approx \frac{C}{2}$



Single link, cross-traffic

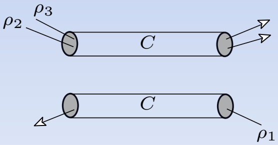
What is the bandwidth share of such a situation ?

- Ack packets get compressed by data packets (which are bigger)
- The download can transmit only half of the time
- Resulting bandwidth of each flow $\approx \frac{C}{2}$



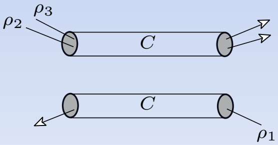
Accounting for cross-traffic is easy with Max-Min

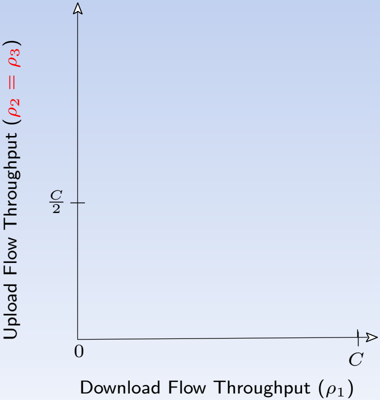
- In the original problem Max-Min give the “wrong” answer



Accounting for cross-traffic is easy with Max-Min

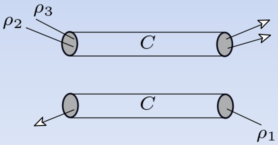
- In the original problem Max-Min give the “wrong” answer





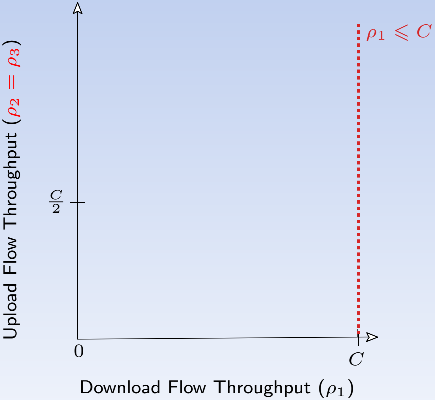
Accounting for cross-traffic is easy with Max-Min

- In the original problem Max-Min give the “wrong” answer



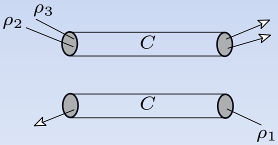
Constraints on Max-Min:

$\rho_1 \leqslant C$



Accounting for cross-traffic is easy with Max-Min

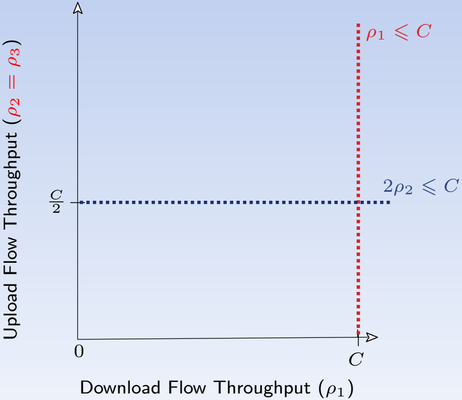
- In the original problem Max-Min give the “wrong” answer



Constraints on Max-Min:

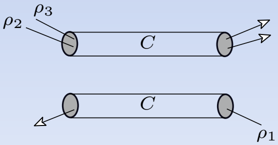
$\rho_1 \leqslant C$

$\rho_2 + \rho_3 \leqslant C$



Accounting for cross-traffic is easy with Max-Min

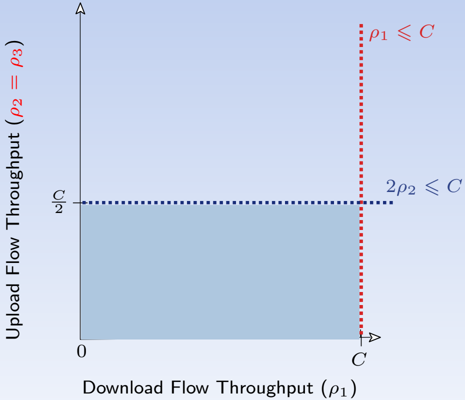
- In the original problem Max-Min give the “wrong” answer



Constraints on Max-Min:

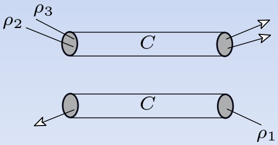
$\rho_1 \leqslant C$

$\rho_2 + \rho_3 \leqslant C$



Accounting for cross-traffic is easy with Max-Min

- In the original problem Max-Min give the “wrong” answer



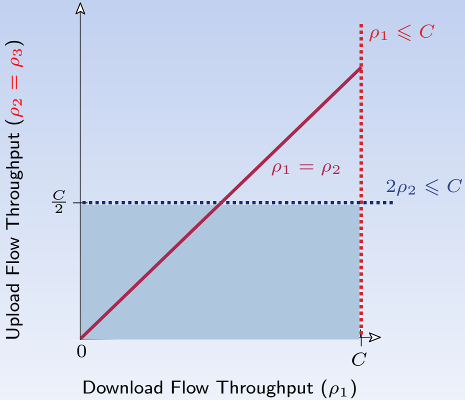
Constraints on Max-Min:

$\rho_1 \leqslant C$

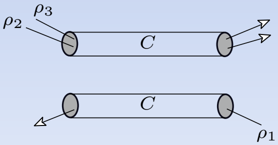
$\rho_2 + \rho_3 \leqslant C$

Objective function:

$\max(\min(\rho_1, \rho_2, \rho 3)) \iff \max(\min(\rho_1, \rho_2))$



- In the original problem Max-Min give the “wrong” answer



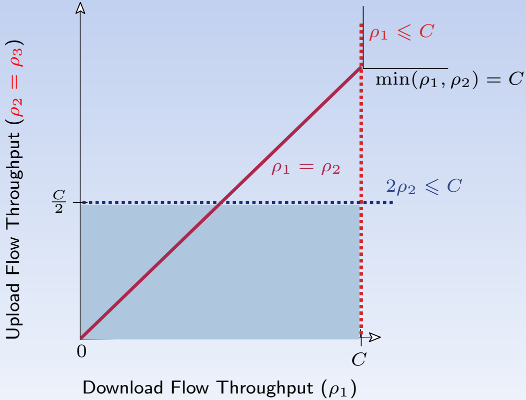
Constraints on Max-Min:

$\rho_1 \leqslant C$

$\rho_2 + \rho_3 \leqslant C$

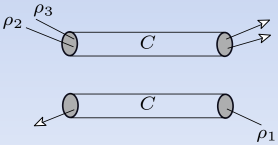
Objective function:

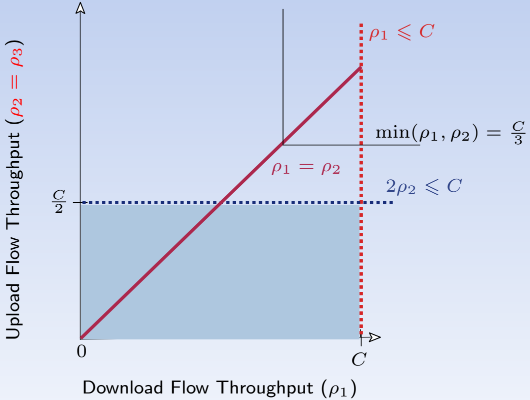
$\max(\min(\rho_1, \rho_2, \rho3)) \iff \max(\min(\rho_1, \rho_2))$



Accounting for cross-traffic is easy with Max-Min

- In the original problem Max-Min give the "wrong" answer





Constraints on Max-Min:

$\rho_1 \leqslant C$

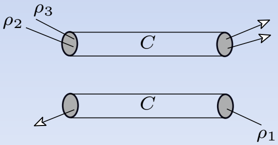
$\rho_2 + \rho_3 \leqslant C$

Objective function:

$\max(\min(\rho_1, \rho_2, \rho 3)) \iff \max(\min(\rho_1, \rho_2))$

Accounting for cross-traffic is easy with Max-Min

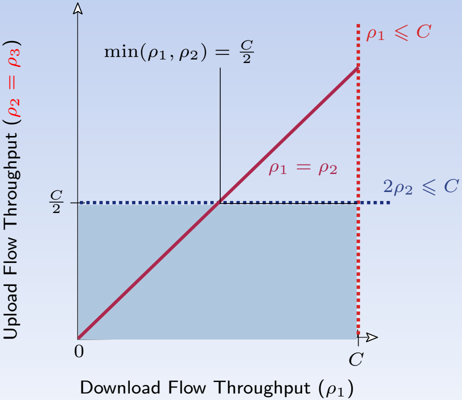
- In the original problem Max-Min give the “wrong” answer



Constraints on Max-Min:

$\rho_1 \leqslant C$

$\rho_2 + \rho_3 \leqslant C$

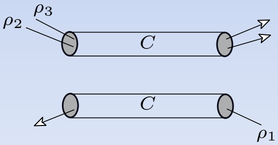


Objective function:

$\max(\min(\rho_1, \rho_2, \rho3)) \iff \max(\min(\rho_1, \rho_2))$

Accounting for cross-traffic is easy with Max-Min

- In the original problem Max-Min give the “wrong” answer



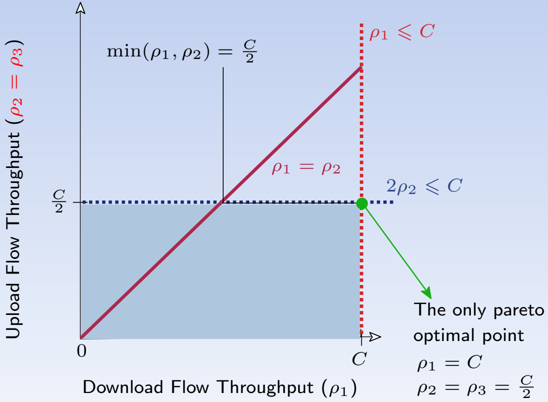
Constraints on Max-Min:

$\rho_1 \leqslant C$

$\rho_2 + \rho_3 \leqslant C$

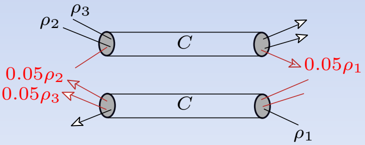
Objective function:

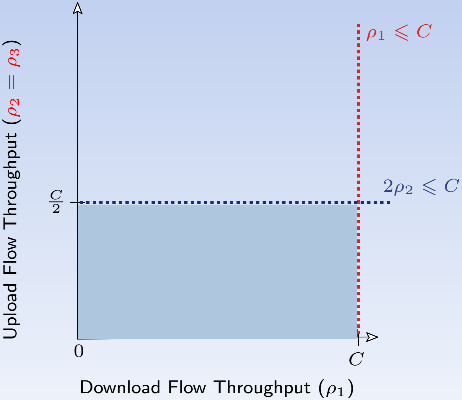
$\max(\min(\rho_1, \rho_2, \rho 3)) \Longleftrightarrow \max(\min(\rho_1, \rho_2))$



Accounting for cross-traffic is easy with Max-Min

- In the original problem Max-Min give the “wrong” answer

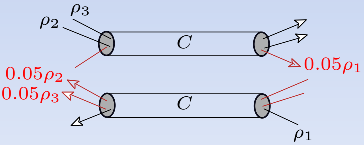


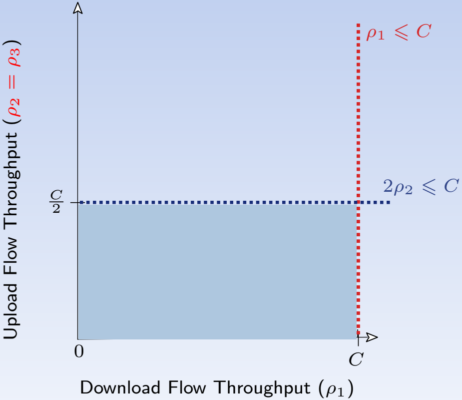


Objective function:

$\max(\min(\rho_1, \rho_2, \rho 3)) \iff \max(\min(\rho_1, \rho_2))$

- In the original problem Max-Min give the “wrong” answer





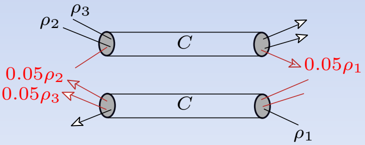
Constraints on Max-Min:

Objective function:

$\max(\min(\rho_1, \rho_2, \rho 3)) \iff \max(\min(\rho_1, \rho_2))$

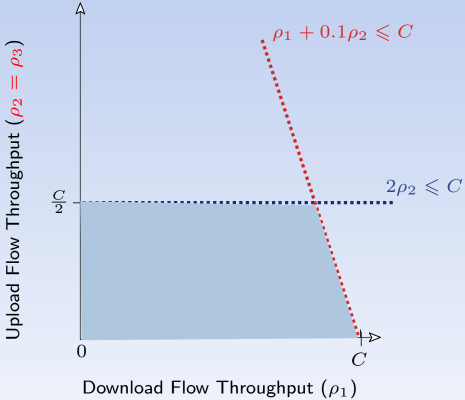
Accounting for cross-traffic is easy with Max-Min

- In the original problem Max-Min give the "wrong" answer



Constraints on Max-Min:

$0.05\rho_3 + 0.05\rho_2 + \rho_1 \leqslant C$

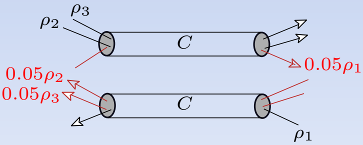


Objective function:

$\max(\min(\rho_1, \rho_2, \rho3)) \iff \max(\min(\rho_1, \rho_2))$

Accounting for cross-traffic is easy with Max-Min

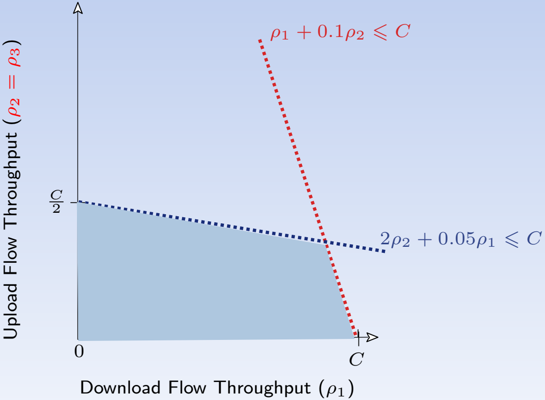
- In the original problem Max-Min give the “wrong” answer



Constraints on Max-Min:

$0.05\rho_3 + 0.05\rho_2 + \rho_1 \leqslant C$

$0.05\rho_1 + \rho_2 + \rho_3 \leqslant C$

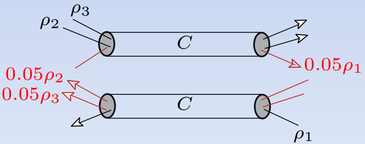


Objective function:

$\max(\min(\rho_1, \rho_2, \rho3)) \iff \max(\min(\rho_1, \rho_2))$

Accounting for cross-traffic is easy with Max-Min

- In the original problem Max-Min give the "wrong" answer



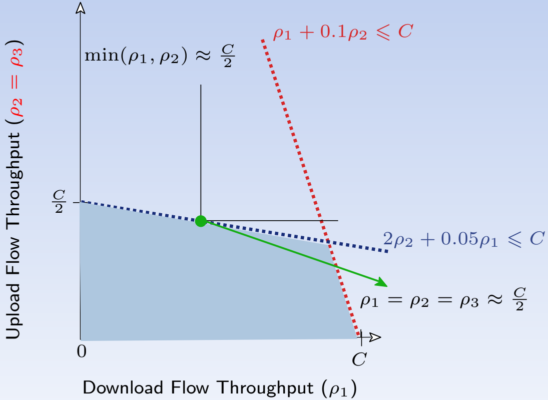
Constraints on Max-Min:

$0.05\rho_3 + 0.05\rho_2 + \rho_1 \leqslant C$

$0.05\rho_1 + \rho_2 + \rho_3 \leqslant C$

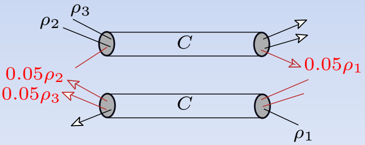
Objective function:

$\max(\min(\rho_1, \rho_2, \rho 3)) \iff \max(\min(\rho_1, \rho_2))$



Accounting for cross-traffic is easy with Max-Min

- In the original problem Max-Min give the "wrong" answer



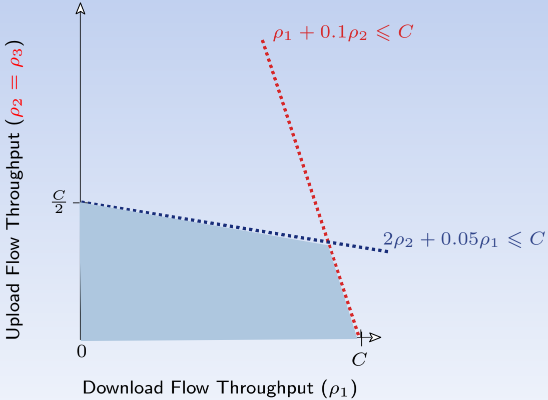
Constraints on Max-Min:

$0.05\rho_3 + 0.05\rho_2 + \rho_1 \leqslant C$

$0.05\rho_1 + \rho_2 + \rho_3 \leqslant C$

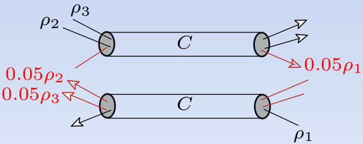
Objective function:

$f(\rho_1, \rho_2) = \log(\rho_1) + 2\log(\rho_2)$



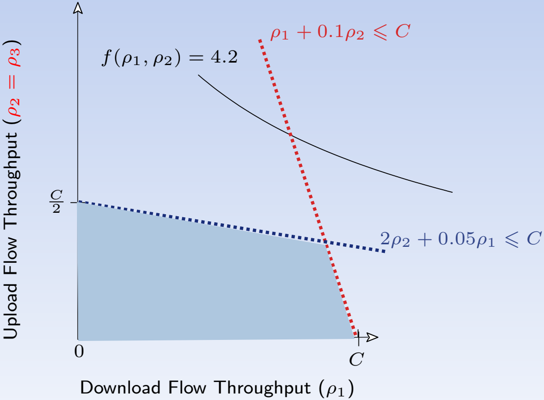
Accounting for cross-traffic is easy with Max-Min

- In the original problem Max-Min give the “wrong” answer



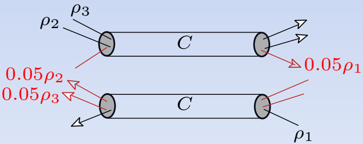
Constraints on Max-Min:
$0.05\rho_3 + 0.05\rho_2 + \rho_1 \leqslant C$
$0.05\rho_1 + \rho_2 + \rho_3 \leqslant C$

Objective function:
$f(\rho_1, \rho_2) = \log(\rho_1) + 2\log(\rho_2)$



Accounting for cross-traffic is easy with Max-Min

- In the original problem Max-Min give the “wrong” answer



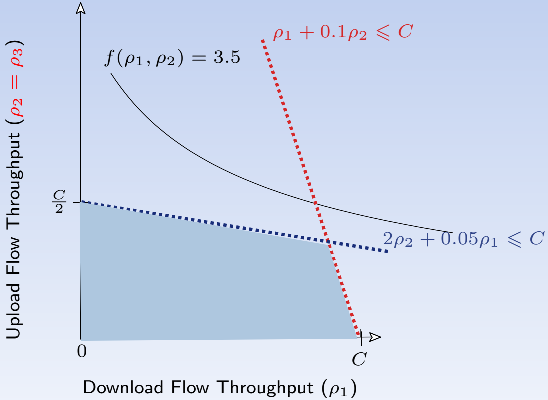
Constraints on Max-Min:

$0.05\rho_3 + 0.05\rho_2 + \rho_1 \leqslant C$

$0.05\rho_1 + \rho_2 + \rho_3 \leqslant C$

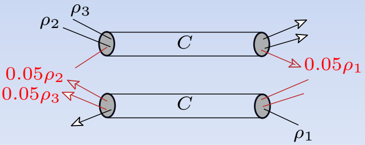
Objective function:

$f(\rho_1, \rho_2) = \log(\rho_1) + 2\log(\rho_2)$



Accounting for cross-traffic is easy with Max-Min

- In the original problem Max-Min give the “wrong” answer



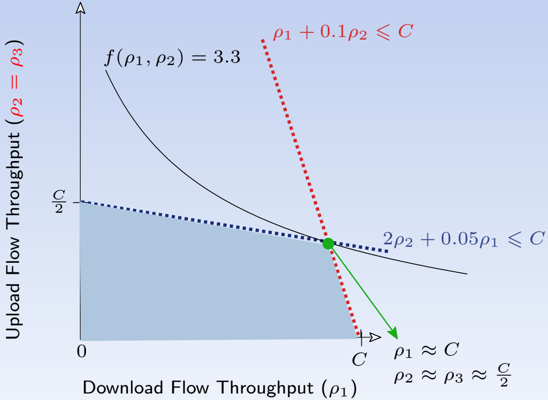
Constraints on Max-Min:

$0.05\rho_3 + 0.05\rho_2 + \rho_1 \leqslant C$

$0.05\rho_1 + \rho_2 + \rho_3 \leqslant C$

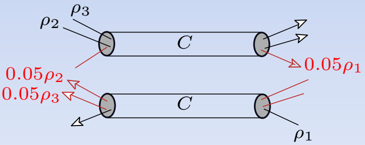
Objective function:

$f(\rho_1, \rho_2) = \log(\rho_1) + 2\log(\rho_2)$



Accounting for cross-traffic is easy with Max-Min

- In the original problem Max-Min give the “wrong” answer



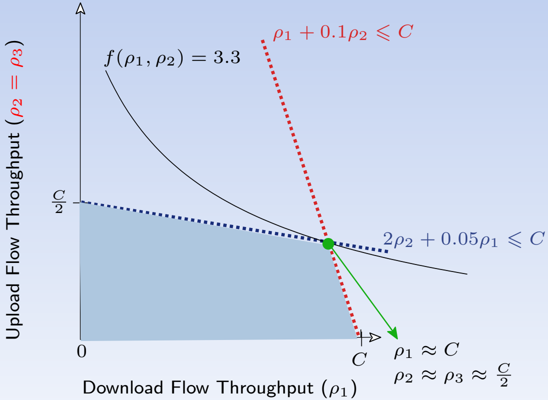
Constraints on Max-Min:

$0.05\rho_3 + 0.05\rho_2 + \rho_1 \leqslant C$

$0.05\rho_1 + \rho_2 + \rho_3 \leqslant C$

Objective function:

$f(\rho_1, \rho_2) = \log(\rho_1) + 2\log(\rho_2)$



This helps neither $\max(\sum(\log(\rho_i)))$ **nor** $\max(\sum(\arctan(\rho_i)))$

Error and limitations of fluid models

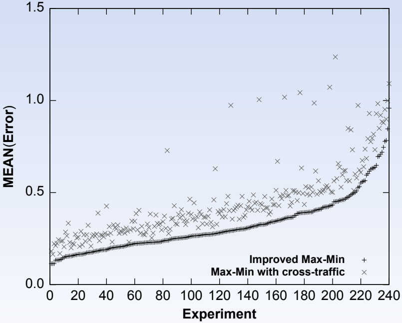
Hypothesis: our new Max-Min model improves validity

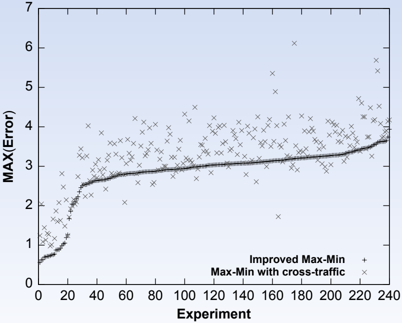
- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

Hypothesis: our new Max-Min model improves validity

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

Comparing improved Max-Min to cross-traffic aware Max-Min



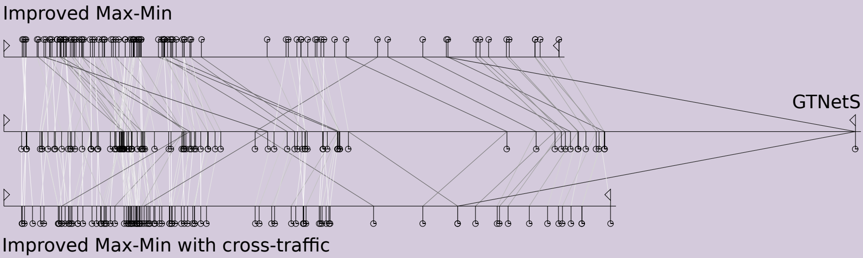


Error and limitations of fluid models

Hypothesis: our new Max-Min model improves validity

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

Comparing completion times instead of bandwidth share



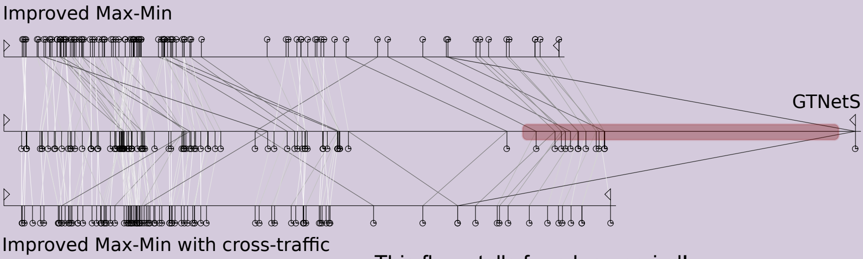
The cross-trafic modification improves prediction for many flows

Error and limitations of fluid models

Hypothesis: our new Max-Min model improves validity

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

Comparing completion times instead of bandwidth share



This flow stalls for a long period!

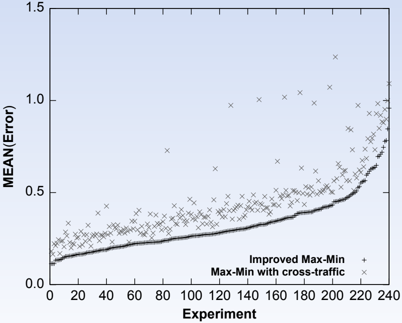
Modeling this with fluid models seems unfeasible

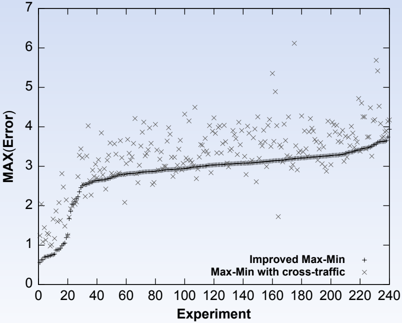
Error and limitations of fluid models

Hypothesis: our new Max-Min model improves validity

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

Comparing improved Max-Min to cross-traffic aware Max-Min



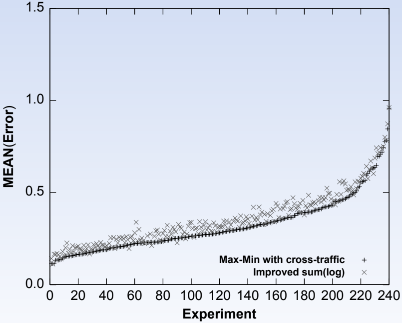


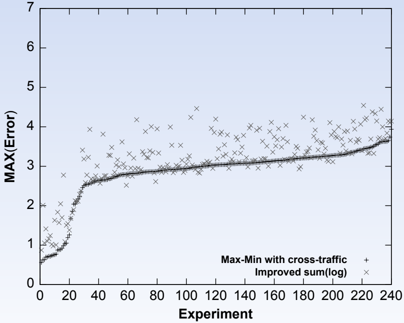
Error and limitations of fluid models

Hypothesis: our new Max-Min model improves validity

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

Comparing cross-traffic aware Max-Min with improved $\sum(\log(\rho_i))$



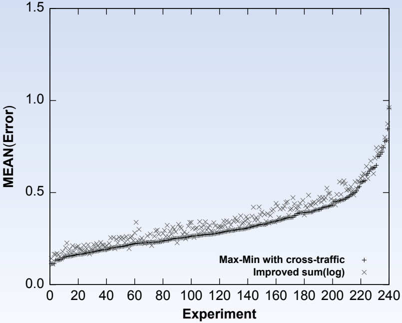


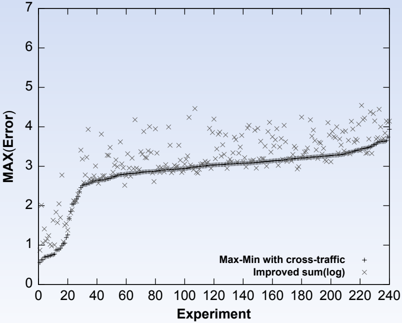
Error and limitations of fluid models

Hypothesis: our new Max-Min model improves validity

- Compare bandwidth sharing for several scenarios
  - 24 random generated platforms
    - 35 to 200 nodes
    - Two random topology models: Waxman and Tiers random
    - Heterogeneous or either homogeneous bandwidth
  - 10 deployments with 100 concurrent flows for each platform

Comparing cross-traffic aware Max-Min with improved $\sum(\log(\rho_i))$





Cross-traffic aware Max-Min has better accuracy

Conclusion & Perspectives

We presented a fair accuracy evaluation of fluid network models

- Most studies try to prove validity by showing situations where the model works
- Models are validated by looking for situations that do **not** work

Conclusion & Perspectives

We presented a fair accuracy evaluation of fluid network models

- Most studies try to prove validity by showing situations where the model works
- Models are validated by looking for situations that do **not** work

Contributions

- Proposed and used a systematic and rigorous methodology for LSDC model validation:
  - Enabled to invalidate very famous TCP models
  - So far, Max-Min is the most accurate fluid model for TCP for our context
  - Publicly available and easily reproducible by others

Conclusion & Perspectives

We presented a fair accuracy evaluation of fluid network models

- Most studies try to prove validity by showing situations where the model works
- Models are validated by looking for situations that do **not** work

Contributions

- Proposed and used a systematic and rigorous methodology for LSDC model validation:
  - Enabled to invalidate very famous TCP models
  - So far, Max-Min is the most accurate fluid model for TCP for our context
  - Publicly available and easily reproducible by others
- This kind of models provides a very good tradeoff between speed and accuracy
- Successfully applied to the Volunteer Computing framework

Conclusion & Perspectives

We presented a fair accuracy evaluation of fluid network models

- Most studies try to prove validity by showing situations where the model works
- Models are validated by looking for situations that do **not** work

Contributions

- Proposed and used a systematic and rigorous methodology for LSDC model validation:
  - Enabled to invalidate very famous TCP models
  - So far, Max-Min is the most accurate fluid model for TCP for our context
  - Publicly available and easily reproducible by others
- This kind of models provides a very good tradeoff between speed and accuracy
- Successfully applied to the Volunteer Computing framework

Perspectives

Conclusion & Perspectives

We presented a fair accuracy evaluation of fluid network models

- Most studies try to prove validity by showing situations where the model works
- Models are validated by looking for situations that do **not** work

Contributions

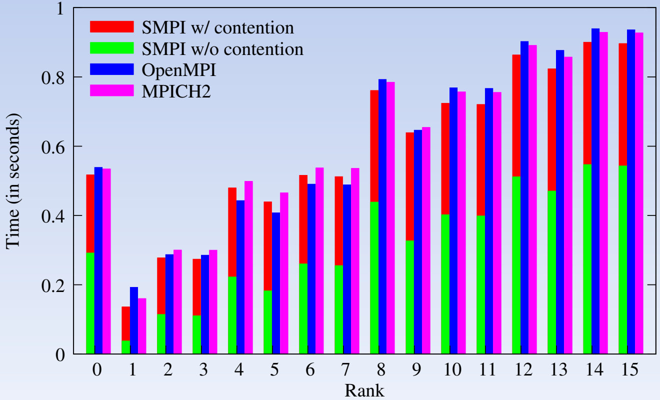
- Proposed and used a systematic and rigorous methodology for LSDC model validation:
  - Enabled to invalidate very famous TCP models
  - So far, Max-Min is the most accurate fluid model for TCP for our context
  - Publicly available and easily reproducible by others
- This kind of models provides a very good tradeoff between speed and accuracy
- Successfully applied to the Volunteer Computing framework

Perspectives

- Apply this methodology to a HPC framework
  - MPI applications
  - High Performance Networks (InfiniBand, Myrinet, . . .)
  - CPU and memory (multicore, NUMA, . . .)
- GPUs, Power consumption, Exascale ?. . .

Questions ?

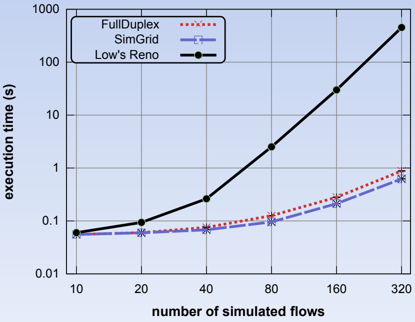
Does contention really matters?



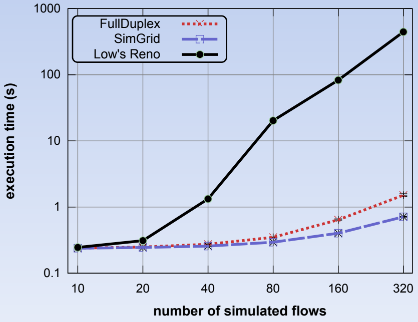
Evaluation of MPI scatter with SMPI by Quinson, Clauss, et al.

What about speed?

- 50 nodes platforms



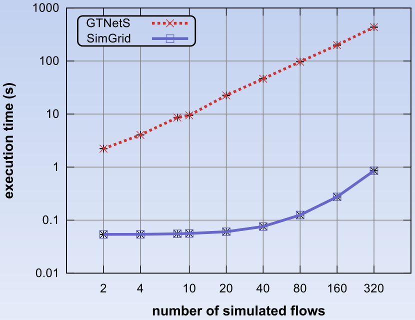
Hierarchical Platform (Tiers)

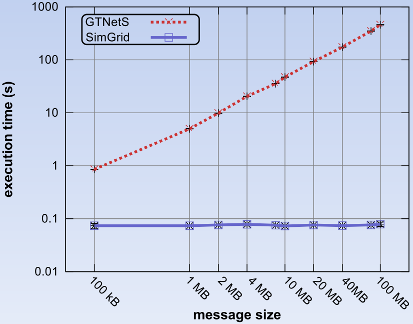


One-level Platform (Waxman)

Our implementation of Max-Min scales well

Model performance compared to GTNetS





Fluid model enables higher scalability than packet-level