



NORTHWESTERN  
UNIVERSITY

# Evaluating Distributed Systems: Does Background Traffic Matter?

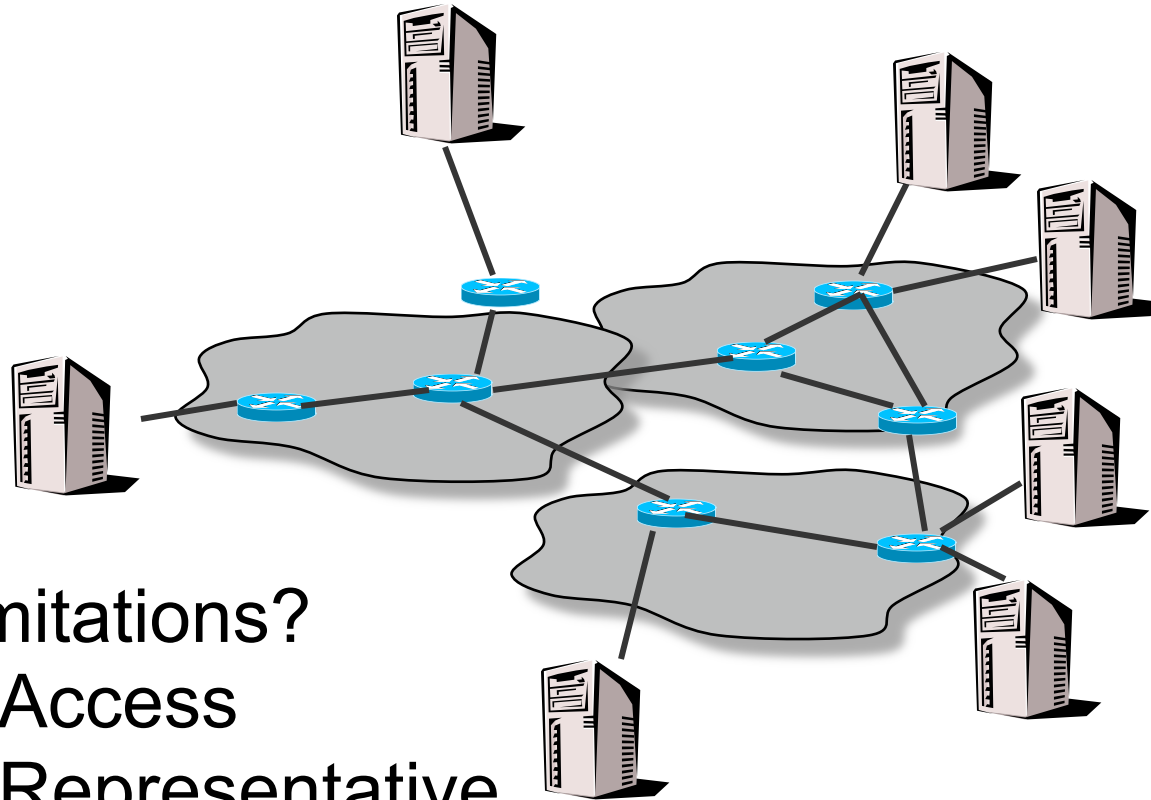
Kashi Venkatesh Vishwanath  
and Amin Vahdat

Instructor: Fabian Bustamante  
Presented by: Mario Sanchez



# The focus of this paper

- Quantify application sensitivity to a range of background traffic characteristics.
- Methodology
  - Present a methodology to deterministically subject application traffic to a range of realistic network conditions while accounting for the complexity of real network traffic.
  - Carry out a systematic sensitivity study using this methodology.

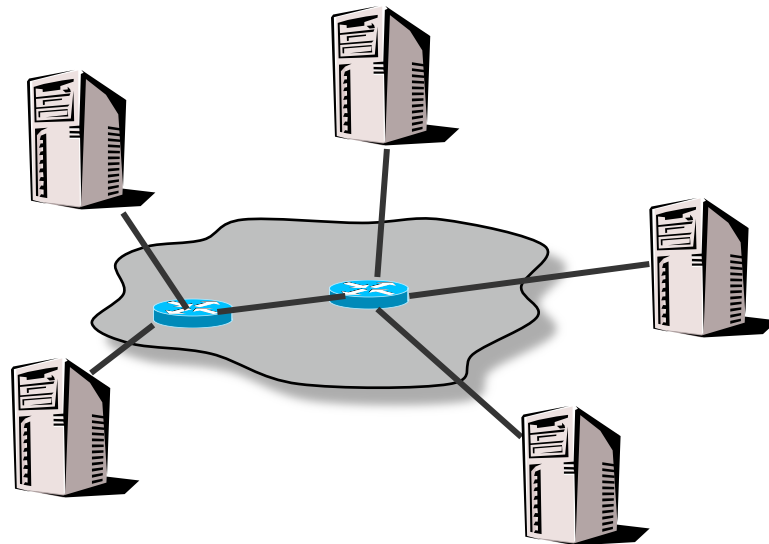
# Straightforward Technique



- Limitations?
- +Access
- +Representative
- +Reproduce

 Router  
 End-Systems

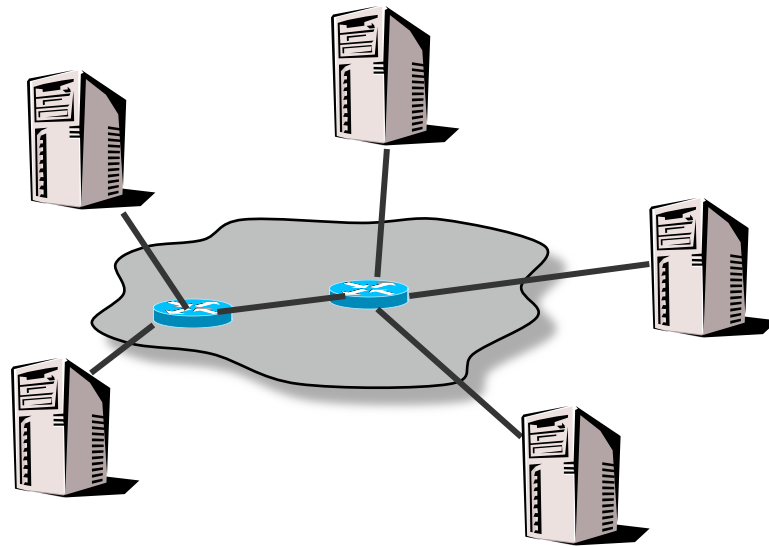
# Local Testbed



- +Simple
- +Tractable
- +Control

What's the catch?

# Local Testbed



+ It doesn't catch the complexity of the Internet

# What would it take to create sufficiently detailed snapshots of the internet?

- Complex task!
- Simpler version of the problem:
  - Focus our attention to one specific link
- Understand:
  - Why does the traffic arriving at the link look the way it does?
  - How does that influence the performance of the applications we are interested in?

# Does background traffic matter?

- Do I need to be able to account for it?
- Is it simply enough to reproduce the average throughput based on the original link that I started with?
- What kind of background traffic model should be used?
- Does the generated background traffic need to be congestion responsive or not?

# Methods used in the community

Explanation	(%)
No Background Traffic	25.6
Constant Bit Rate Traffic	2.33
Fixed Loss Rate	2.33
Lowered Link Capacity	2.33
Only Latencies	2.33
“Some” Background Flows	2.33
“Some” TCP source	2.33
Custom Built Simulator	4.65
Pareto Flow Arrival	4.65
Fixed Length Flows	2.33
Long Lived flows	4.65
LRD Traffic	2.33
Pareto Length Flows	2.33
SpecWeb	6.98
Run on PlanetLab	18.6
Real World Deployment	9.3
Harpoon	2.33
RON Testbed	2.33

No models (1/4)

Simple models (1/6)

35 papers, 43 experiments

Sophisticated models (1/4)

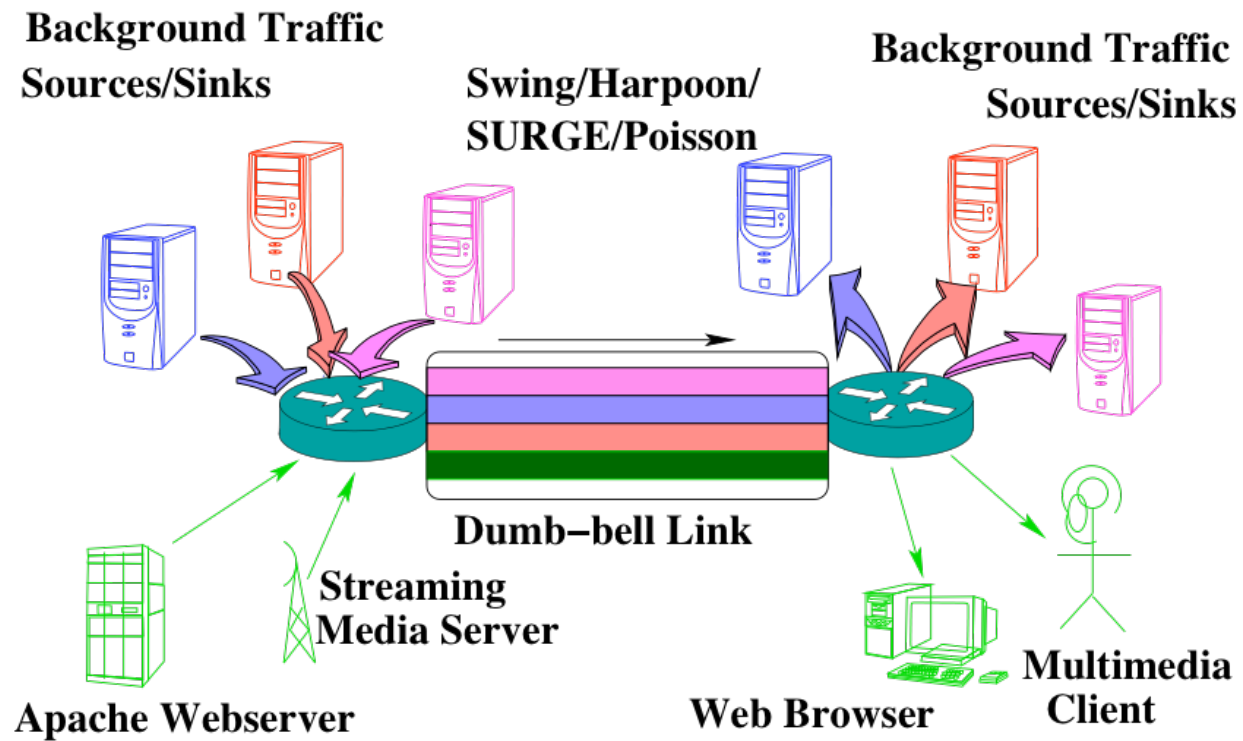
More realistic environment (1/3)



# Methods used in the community

- ~ Lack of understanding
- ~ Only one single model was used, no sensitivity analysis to the background traffic model used.

# Architecture



# Choices for BGT generator

Model traffic based on long-term averages:

1. At a constant rate (**CBR**): generate packets at a specified rate from sources to sinks.
2. According to a **Poisson** arrival process: with Poisson byte arrival per unit time

# Choices for BGT generator

3. Take a trace of traffic at a particular point in the Internet and replay packet timings:
  - Using UDP as transport layer:
    - Can be fairly realistic in reproducing traffic characteristics.
    - Generated traffic will be completely agnostic to the presence of application traffic; thus some of the desired responsiveness will be lost.

# Choices for BGT generator

- Using TCP as transport layer:
  - Generated traffic clearly will interact with application traffic, giving back some of the responsiveness.
  - Reproduced timing of original packets is no longer assured (due to congestion control mechanisms); thus some amount of realism will be lost.

# Choices for BGT generator

## 4. More Sophisticated Models:

- Realistic: Look like the background traffic measured from background traffic link.
- Responsive: It should really be able to interact with the application traffic exactly like it does on the internet, instead of killing it.
- Flexible: flexible enough that projected scenarios into the future can be expressed.

## **SWING traffic generator**

# Swing traffic generator

- Observe packets entering and exiting a link:
  - Time between requests
  - Application behavior
  - Distribution of request/response sizes
- Build a model that explains the traffic process for the link (bandwidth, latencies, loss rates)
- Configure a set of machines which talk to each other using real TCP packets and reproduce statistically similar traffic.

# Different Applications

- Web Traffic
- Multimedia
- Bandwidth Estimation Tool
  
- Apart from **throughput** we are also interested in **burstiness** properties

In the interest of time I'm going to skip some of these...

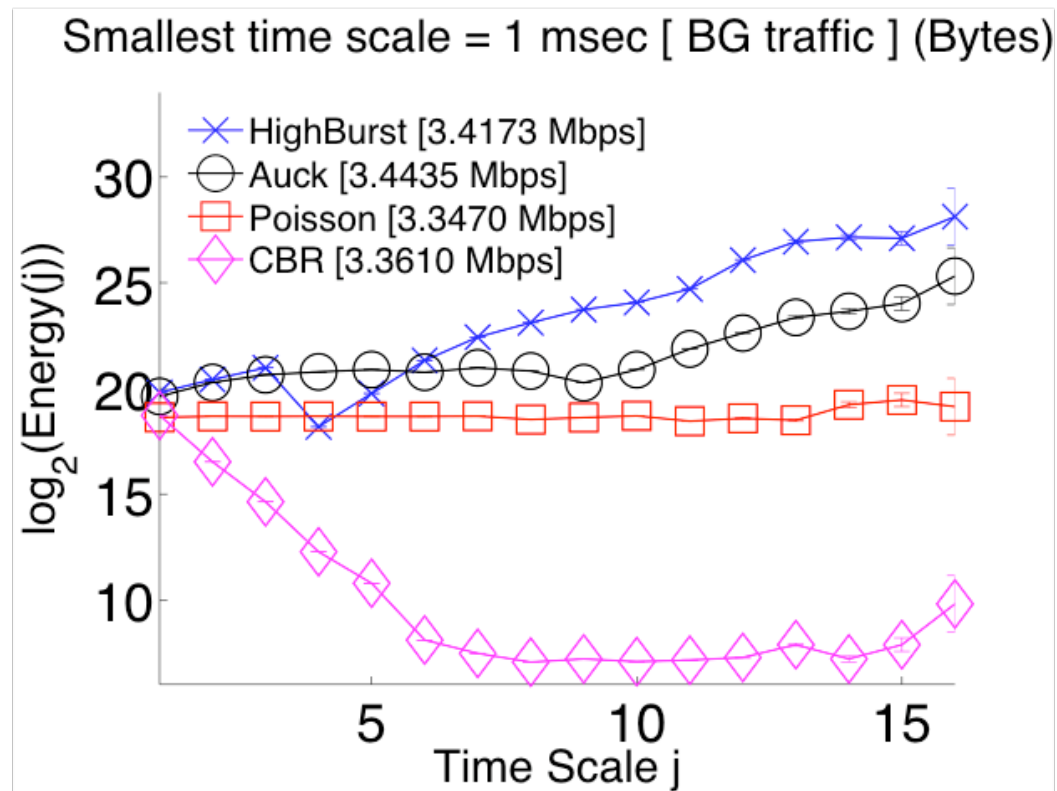


# Traffic Traces

Trace ↓	Secs	Trace BW		Trace Collection Date	Number of flows	Dominant applications	Unique IPs (1000s)
		Aggregate (Mbps)	Dir0 (Mbps)				
Auck	600	5.5	3.3	June 11, 2001	155 K	HTTP, SQUID	3
Mawi	900	17.8	7.8	September 23, 2004	476 K	HTTP, RSYNC	15
Mawi2	900	11.9	10.8	December 30, 2003	160 K	HTTP, NNTP	8

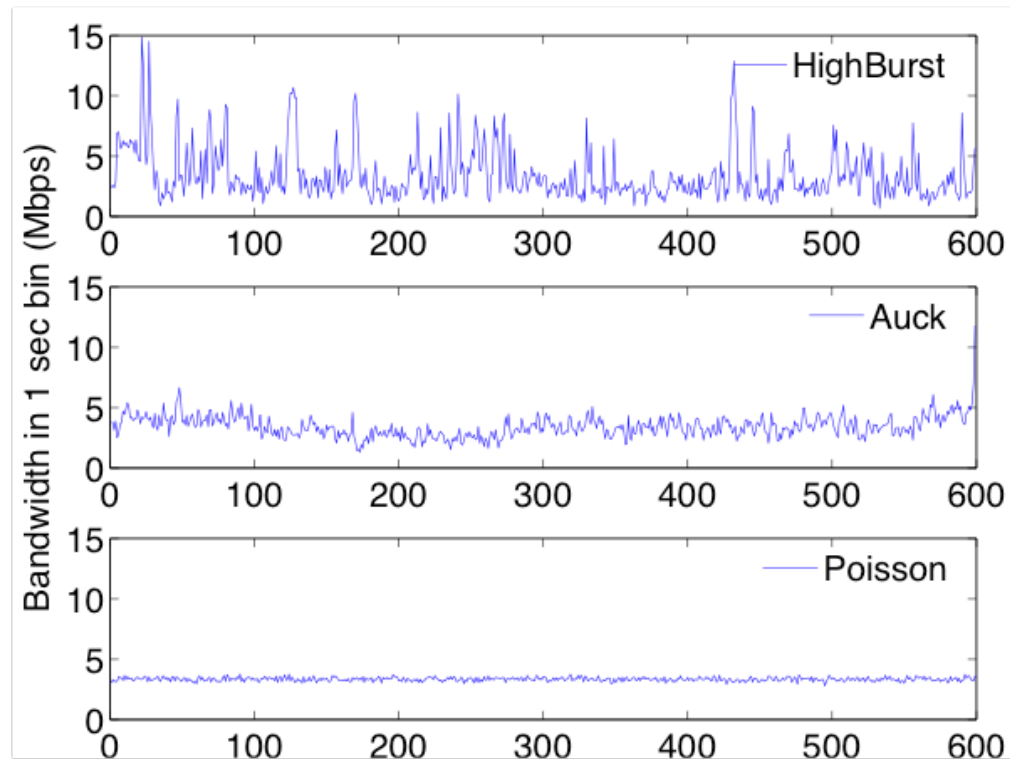
- Picked based on difference in time, geographical proximity and variation in throughput numbers.

# Wavelet-scaling/energy plot



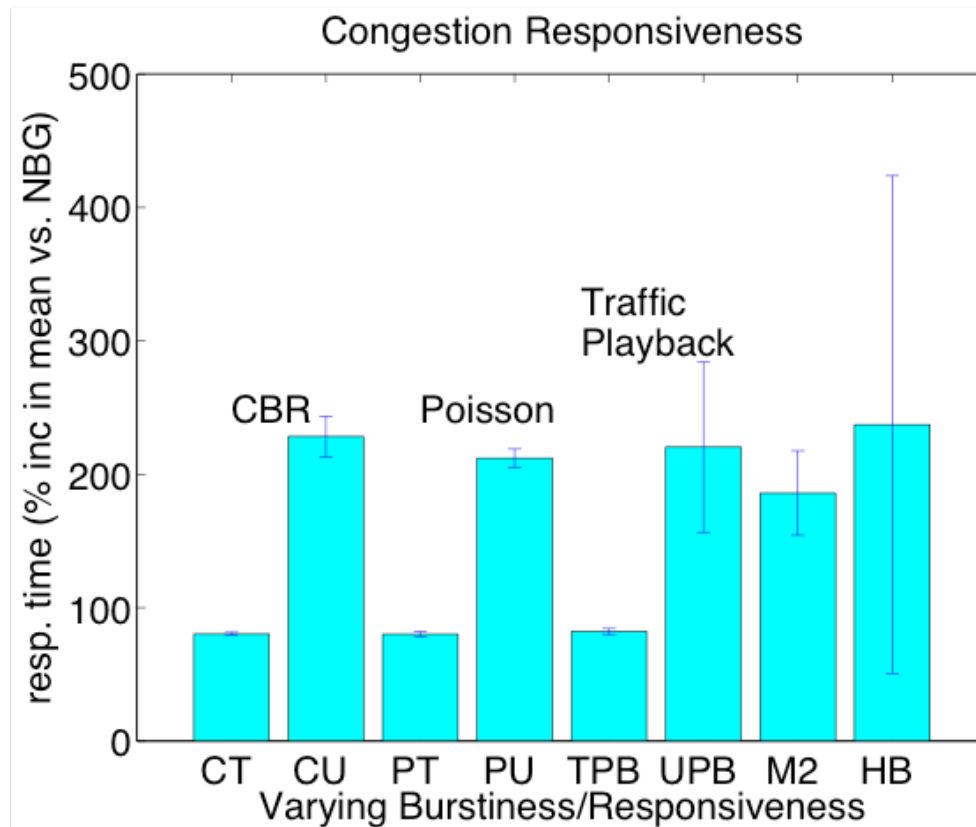
\* Auck-based background traffic

# Time Series



\* Auck-based background traffic

# Experiments: Httpperf/Apache



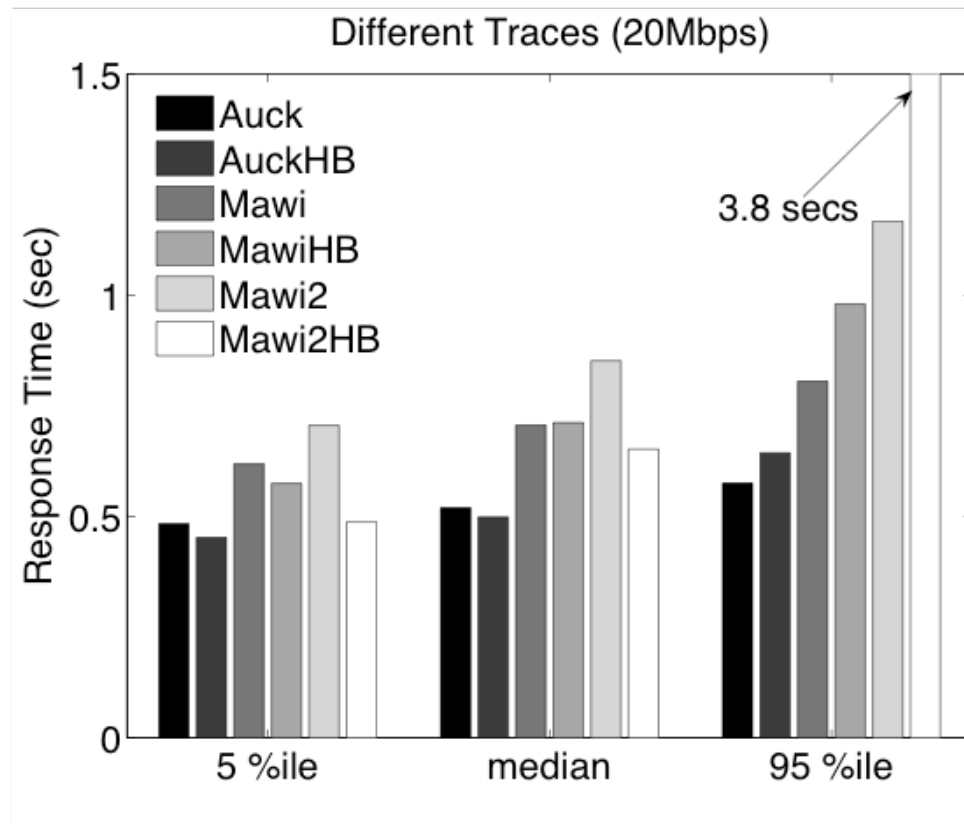
\* 1MB file download, 15Mbps link

# Takeaways

- Congestion-responsive background traffic generators should be used to avoid give away of large portion of the link (fairly contention).
- Simple models do not reproduce the results that we are interested in if we played a realistic traffic model instead.
- Background traffic is important to consider!

Is simply replaying some sort of bursty background traffic enough?

# Bursty Traffic is Not Enough



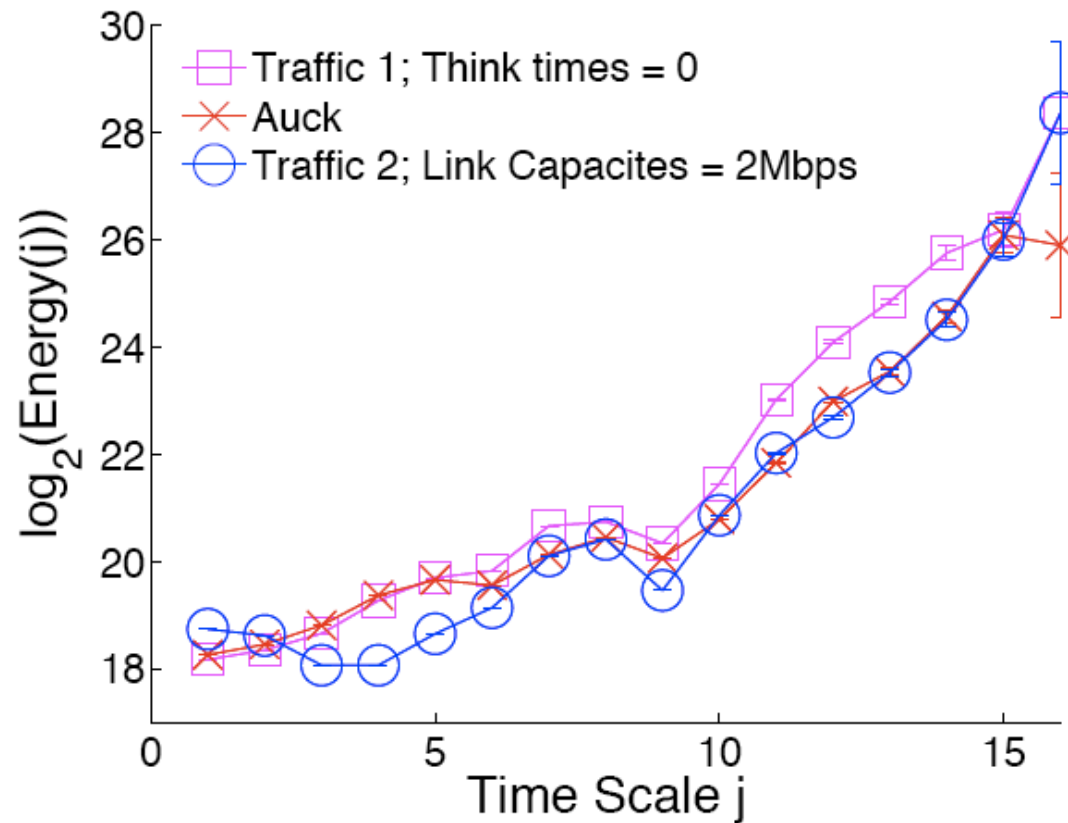
\* 1MB file download, 20Mbps link

# Takeaways

- It is not just the throughput but also the average burstiness of background traffic that matters.
- It is extremely difficult to predict a priority by simply looking at the trace and the application given at hand that why should it matter or not.

“Individual applications have definitive time scales, if background traffic happens to vary across those timescales then the apps are going to see it otherwise they will not.”

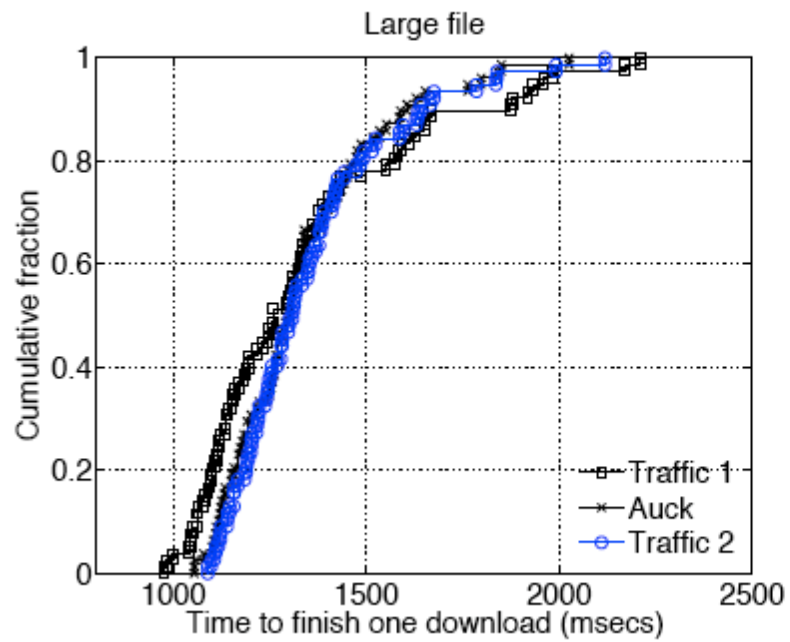
# Burstiness of BGT models



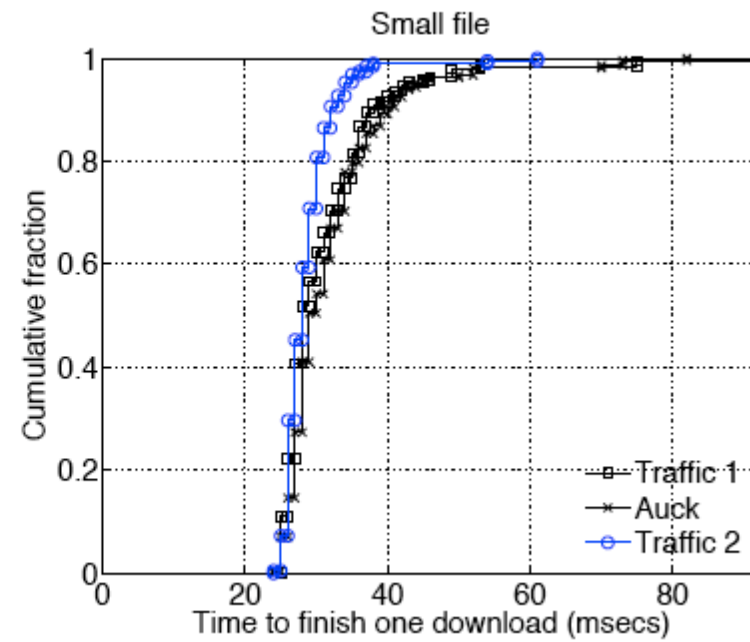


# Comparison

Avg download time = 1sec



Avg download time = 32ms



\* CDFs of download times

# Takeaways

- Individual applications have affinities to particular timescales; if you happened to vary the burstiness of background traffic in that timescale the app will see but otherwise it will not.

# Overview of Results

- Accounting for background traffic is critical to understanding the performance
- Each traffic generator used should have 2 key properties:
  - It should at least look like the original link used to obtain the trace from, so it should be **realistic**.
  - It should match not only the **throughput** numbers but **burstiness** properties that is inherent in internet-like traffic at different time scales.

# Overview of Results

- The generated traffic should be congestion-responsive.
- Certain applications are fairly sensitive even to small perturbations in the characteristics of background traffic while others are completely agnostic.

# Comments

- The study sort-of states the obvious: background traffic is important but...
- Don't give descriptive suggestions based on whether or not to use background traffic and what models to use.

QUESTIONS?

Thank You