A multi-level TCP model with heterogeneous RTTs

Pasi Lassila[1] and Michel Mandjes[2]

[1]Networking Laboratory (HUT)

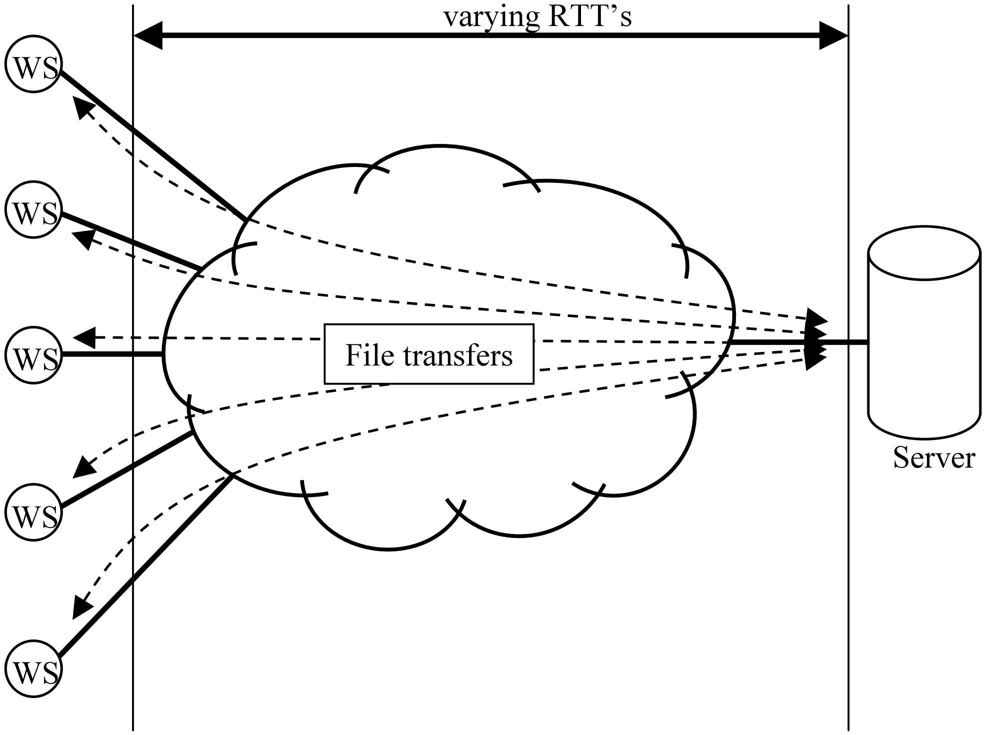
[2]University of Twente / CWI (the Netherlands)

Motivation

- Most Internet traffic carried by TCP
- Main performance measures: throughput and delay

Scenario

- Requests arrive randomly, files have random lengths
- Issues: packet losses, RTTs
- Bottleneck(s): access link, network, server link



Purpose

- Understand how above affects delay performance
- Quantify the dependencies

Earlier work

- Paper presented at ITC-18 (Berlin, August 31 - September 5, 2003)
- Simplified scenario
  - captures sending rate limitation, one bottleneck link (losses), …
  - … but can account for only one RTT!
- Main interest: mean delay
  - Slow start compensation heuristic expressed in a way that requires file lengths to be long enough that TCP steady state is reached
  - ⇒ Model for relatively long file transfers (depending on the bandwidth delay product)
- Results promising, but applicability limited

Present study

- We only consider averages (mean values)
- Main objective: relax assumption of a single RTT
  - Flows with different RTTs share the capacity such that the flows with smaller RTT get more throughput than flows with larger RTTs
  - Assumption: flows are grouped into classes according to RTT
- Include effect of access rate limitation
- Express initial slow start effect such that files of any size can be handled
  - Short files (web mice) simply never leave slow start and will never reach TCP “steady state”

Other approaches

- Fayolle [1980] model: DPS (Discriminatory Processor Sharing), idealized model where flows share the capacity in a weighted manner
  - does not include effect of rate limitations
  - Bu & Towsley [2001] have used the Fayolle model to consider different RTTs
- Ayesta et al. [2003] have considered the conditional mean delay (given the file size) of short flows (web mice)
  - do not explicitly take into account rate limitations, but study the effect of rate limitations on accuracy of Poisson arrival assumption at the buffer (and packet loss estimates)
  - basically only use their model in load scenarios where bottleneck sharing does not occur
- Extending the earlier GPS model difficult
  - Possible to make assumptions under which everything is Markovian
  - Can be generalized to multiple classes and one can (in theory) contruct the generator of the multidimensional process
  - Computationally too intensive for any realistic system

Modeling approach: step 1

- Consider a single link and one RTT
- Assume files lengths exponentially distributed with mean $1/\mu$
- Model the time evolution of the mean number of flows in the system, $N(t)$, and the mean sending rate of the TCP aggregate, $\lambda(t)$

$$\begin{cases} \dfrac{dN(t)}{dt} = \nu - \mu \overbrace{\lambda(t)\big(1-P(t)\big)}^{C(t)}\big(1-\pi_0(t)\big) \\ \dfrac{d\lambda(t)}{dt} = \big(1-P(t)\big)\dfrac{N(t)}{R^2} - P(t)\dfrac{\lambda^2(t)}{2N(t)} \end{cases}$$

  - Idea: use the ideal PS model, but with a reduced goodput, $C(t)$; goodput reduction determined by TCP’s dependence on RTT and packet loss
  - π_0(t) given by a quasi-stationarity approximation of the corresponding PS system
  - $P(t)$ given by the packet loss probability in an M/G/1/K queue with arrival rate $\lambda(t)$

Modeling approach: step 2

- Model for single link case with *M* classes each with own RTT
- Assume that all classes have a common mean file length $1/\mu$
- With Poisson flow arrivals, the total number of flows in the system still behaves as in a PS system for any work conserving service discipline
  - Mean number of flows in each class is divided in proportion to the goodput share of each class
  - Goodput of the system, *C*(*t*), takes into account the sending rate obtained by each TCP class (captures different RTTs)
  - Classes share the bottleneck bandwidth such that link is full

$$\begin{cases} \dfrac{dN(t)}{dt} = \sum_i \nu_i - \mu \overbrace{\sum_i \lambda_i(t)\big(1 - P(t)\big)}^{C(t)}\big(1 - \pi_0(t)\big) \\ \dfrac{d\lambda_i(t)}{dt} = \big(1 - P(t)\big)\dfrac{N_i(t)}{R_i^2} - P(t)\dfrac{\lambda_i^2(t)}{2N_i(t)} \qquad , i = 1, \ldots, M \end{cases}$$

$$N_i(t) = \frac{\nu_i / \lambda_i(t)}{\sum_i \nu_i / \lambda_i(t)} N(t)$$

Modeling approach: step 3

- Effect of rate limitation
  - Observation: Each flow is only limited by its sending rate as long as number of ongoing flows is less than it takes to fill the bottle neck link ⇒ M/G/∞ model. When this point is exceeded the system switches to processor sharing mode.
  - Which operating region is reached is determined by comparing the PS system sending rate estimate to the actual sending rate limit

$$r_i = \min\left(\lambda_i^{\max}, \lambda_i (1 - P) / N\right)$$

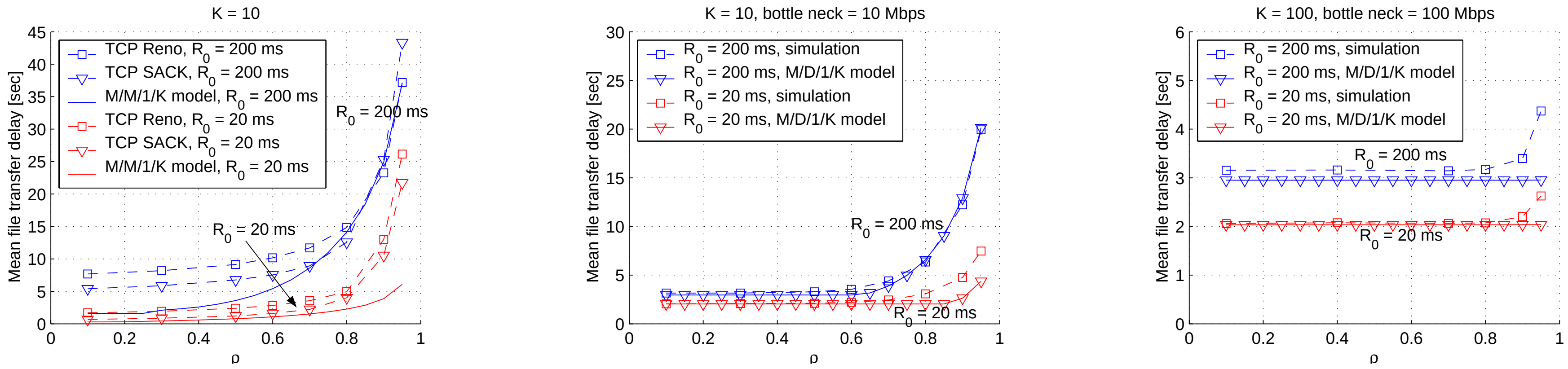
- Mean delay, D_i, consists of
  - length of slow start (time it takes to reach estimated sending rate) +
  - time to send remaining file at the estimated sending rate
- Mean number of flows:
  - If ri determined by the PS limit, then mean number of flows equals N
  - If ri determined by sending rate limit, then mean number of flows in an M/G/∞ model with arrival rate ni and mean service time D_i

Validation

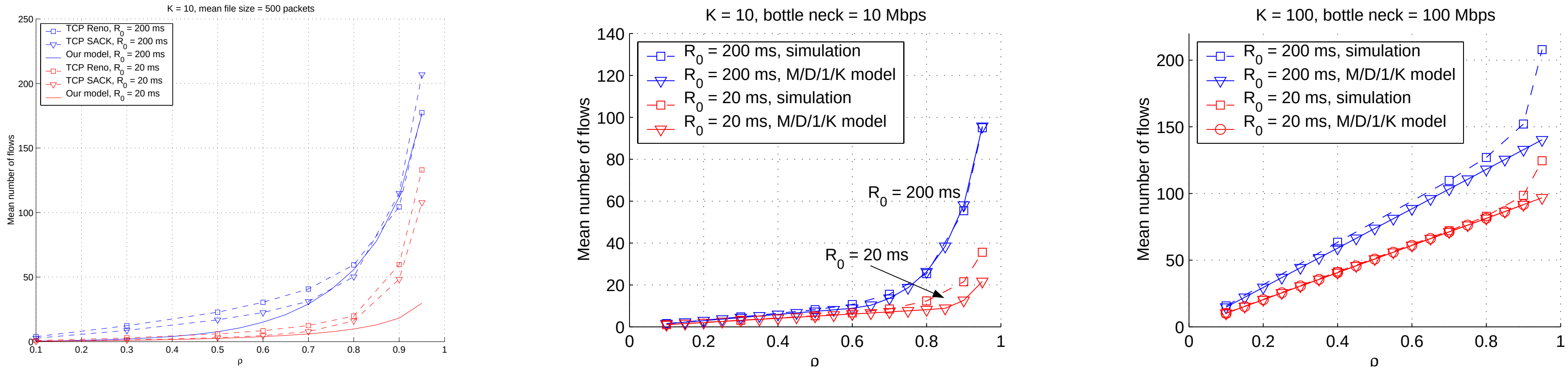
- Validation concerns only the steady state results (no dynamics)
- Tests with:
  - different TCP versions (Reno, Sack)
  - access speed / bottle neck speed ratios
  - different buffer sizes
  - different RTTs
  - different file size distributions
  - different queuing models (M/M/1/K, M/D/1/K)
- Results on
  - mean number of customers
  - mean delays

Single RTT tests: effect of access rate limitation

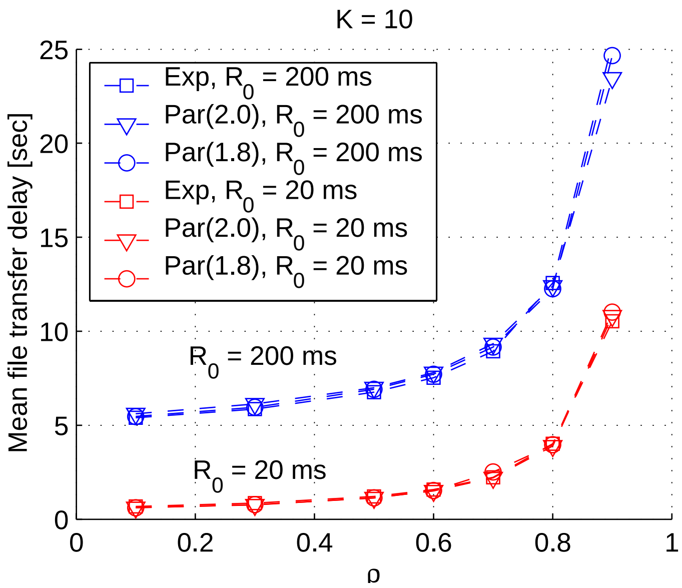
Mean delays

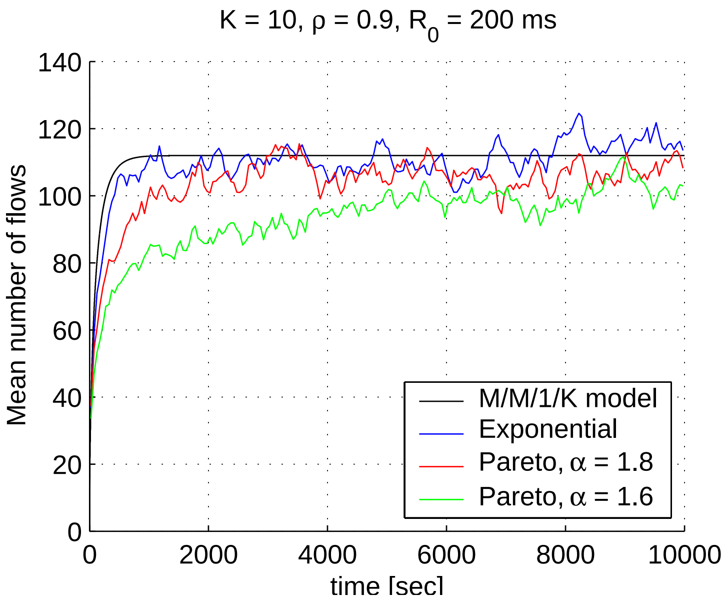


Mean number of flows

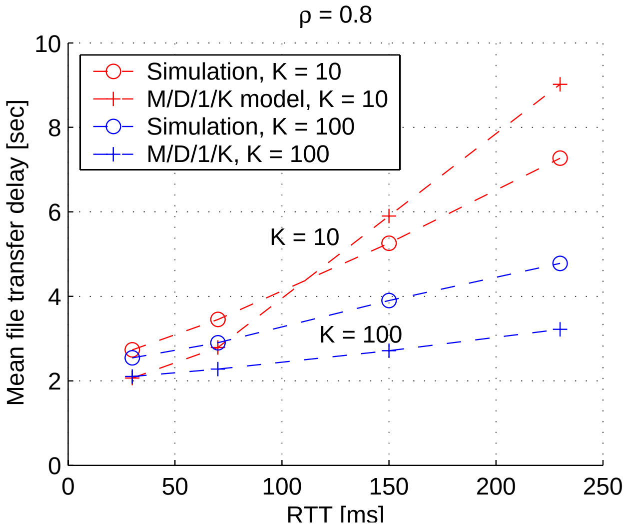


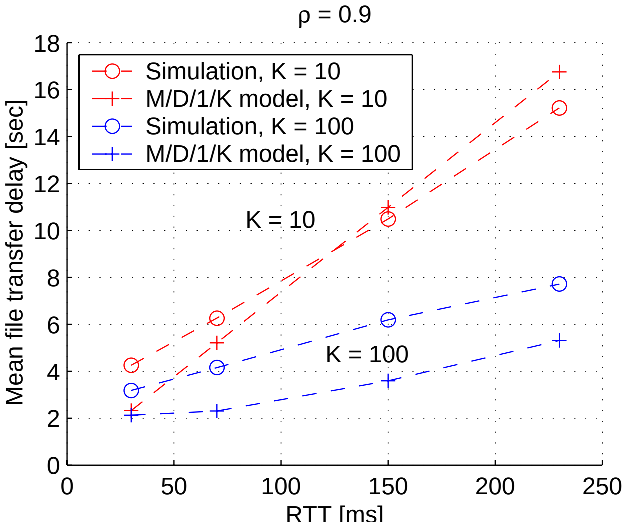
Single RTT tests: insensitivity and dynamics

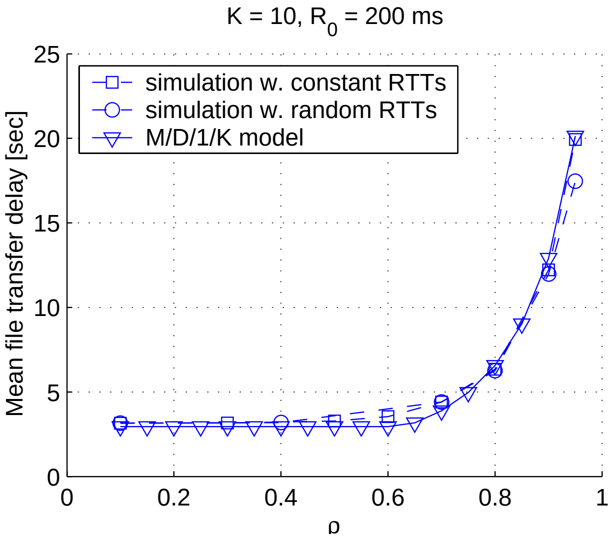




Multiple RTT classes and random RTTs







- RTTs ~ U(10,390) ms
- drawn independently for each file transfer
- mean delay still almost the same as with constant RTT of 200 ms

Conclusions

- Model for mean delays of TCP sources sharing a single bottleneck
  - Captures unequal sharing due to different RTT classes,
  - Effect of access rate limitation, and
  - Initial slow starts
- Results are qualitatively correct but accuracy depends on parameters
  - Typically more accurate as the ratio of access bandwidth to the bottleneck bandwidth decreases ($\Rightarrow$ more multiplexing of TCP sources)
- Open issues
  - Effect of retransmission timeouts, especially during slow start
    - We assume that TCP operates perfectly according to AIMD without time outs
  - Packet loss model (M/D/1/*K*) not very accurate
    - Packet arrival process is actually more like a batch arrival process (instead of Poisson)