“*Network Fairness for Heterogeneous Application*”

Dah Ming Chiu and Adrian S. W. TAM
Department of Information Engineering
The Chinese University of Hong Kong

Content

- Definition of heterogeneity
- Problem of heterogeneous application
- Current solution
- Our proposal
- Simulation and Results

Internet application:

- Elastic traffic
  - Commonly in TCP (protocol 6)
  - Transfer a finite bunch of data over the network
  - Aim: As soon as possible
  - It is useful only after the last bit data is sent
  - Example: downloading a file

Internet application

- Inelastic traffic
  - Commonly in UDP (protocol 17)
    - some protocols (e.g. RTP) may run on top of it
  - Aim: Keep using a constant amount of bandwidth
  - Think of it shows you a movie, it is useful when it gets the bandwidth requirement satisifed
  - It finishes sending when the movie is end

Characteristics

- Elastic traffic is elastic by doing congestion control
  - It shrinks and grows to adapt with the available bandwidth
- Inelastic traffic do not do congestion control
  - It sends with standard rate constantly
  - If the link is overloaded, its packet will drop without retransmission

Problem

- If there are significant number of inelastic traffic
  - Most bandwidth is consumed by inelastic
  - Packets are drop heavily due to inelastic traffic
  - Elastic traffic react to drops and its rate reduced

- Result:
  - Elastic traffic use <u>tiny</u> amount of bandwidth
  - Inelastic traffic use <u>majority</u> of bandwidth
  - It is **not fair**

Current Solution: TCP-friendly Congestion Control

- Ask inelastic traffic do congestion control as well
  - When you see congestion, react by *shrinking* your usage
  - When you see the congestion relieved, *bounce back*

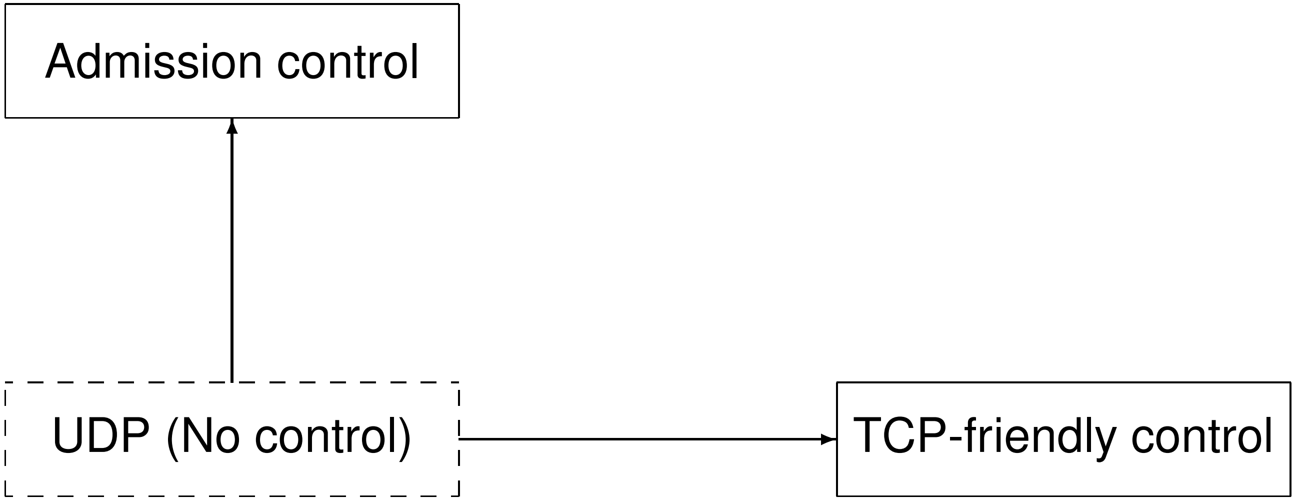
- Widely-accepted solution, but not widely adopted
  - RealPlayer: Choose your rate manually before you start

- TRFC by Padhye (RFC3448) is one of the TCP-friendly solution

- But: Everything can be shrinked and growed at anytime?

Not everything can be shrinked and growed at anytime

- TCP-friendly is not suitable in those cases
- TCP-friendly is not the only solution
- We propose to have *admission control* as an alternative solution



Admission control

UDP (No control)

TCP-friendly control

Idea of Admission Control

- Distributed, not Centralized

- We have parameters α and ε such that
  - Every inelastic user will use α of the bandwidth
  - But upon using the network, they make sure every elastic flows can get a share of at least ε of the bandwidth
  - Typical value: $\alpha = 0.05$ and $\varepsilon = 0.001$

- Admission function:

$$n\varepsilon + (m+1)\alpha \leq 1$$

where $n =$ Number of elastic user and $m =$ Number of inelastic user

Idea of Admission Control

$$n\varepsilon + (m+1)\alpha \leq 1$$

- The new inelastic flow is admitted only if this is true
- If the flow is not admitted, it will tell the user that the network is not suitable to continue and then quit
  - Without using *any* network resource afterwards
- Question:
  - Is this a good way to do?
  - Would this be better than uncontrolled use?
  - How to compare adm control with cong control?

Our approach:

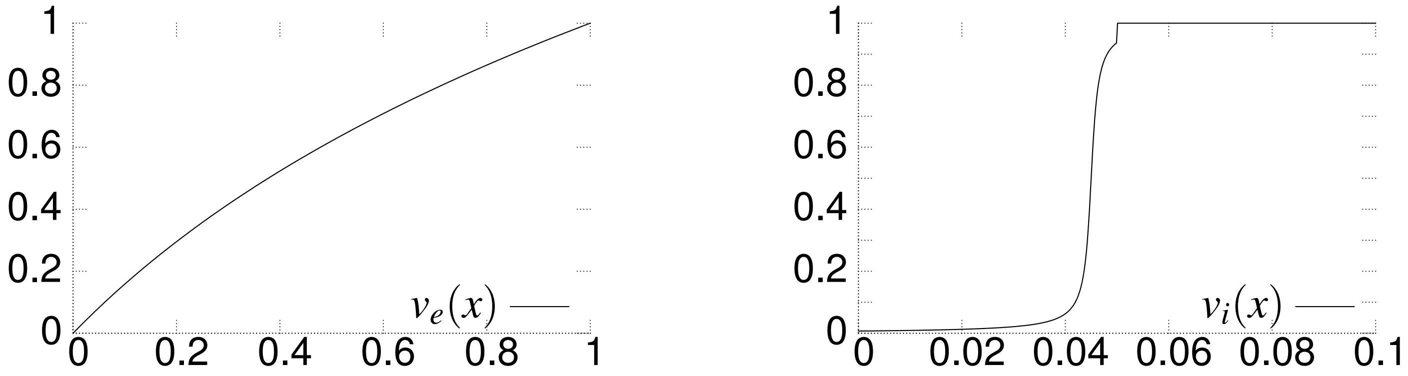
- Formulate the system, and form a 2-class Markov model
  - System state: (n,m) to mean n elastic and m inelastic users in the system
- Simulation by using the Markov model

We need to compare how “good”, hence a utility function is introduced:

- Elastic utility: $U_E = \rho_e \sum_{\substack{(n,m)\\ n\neq 0}} \ln\left(1+(e-1)a_e(n,m)\right) P(n,m|n\neq 0)$
- Inelastic utility: $U_I = (1-B)\alpha\rho_i \sum_{\substack{(n,m)\\ m\neq 0}} \left(\frac{1}{\pi}\arctan(\gamma(a_i(n,m)-\beta\alpha))+\frac{1}{2}\right) P(n,m|m\neq 0)$

We define $\frac{1}{\pi}\arctan(\gamma(\alpha - \beta\alpha)) + \frac{1}{2} = 1$, and the following are the charts of:

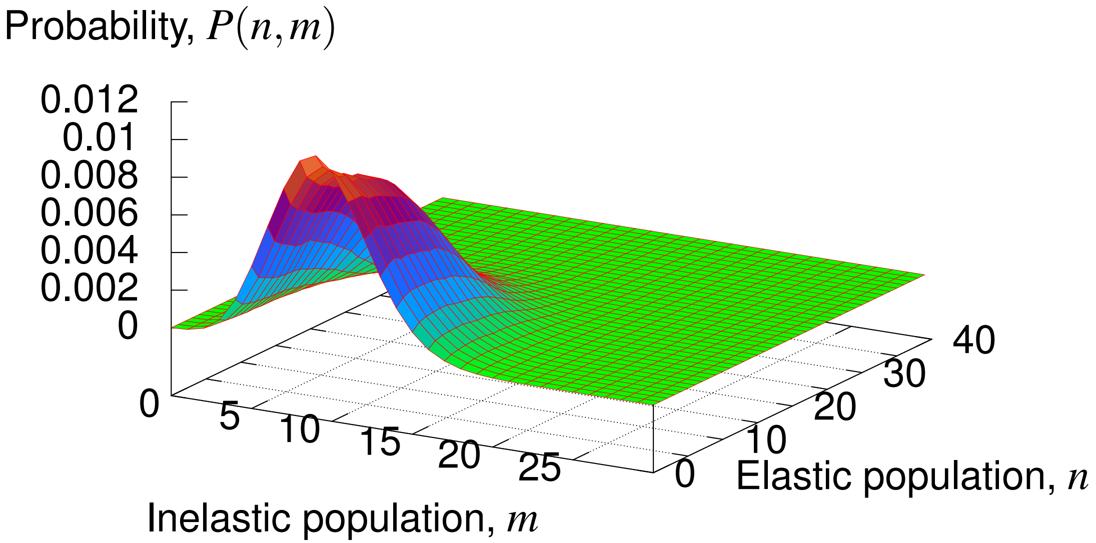
- $v_e = \ln(1 + (e-1)x)$
- $v_i = \begin{cases} \frac{1}{\pi}\arctan(1000(x - 0.9 * 0.05)) + \frac{1}{2} & \text{if } x < 0.05 \\ 1 & \text{otherwise} \end{cases}$



Results

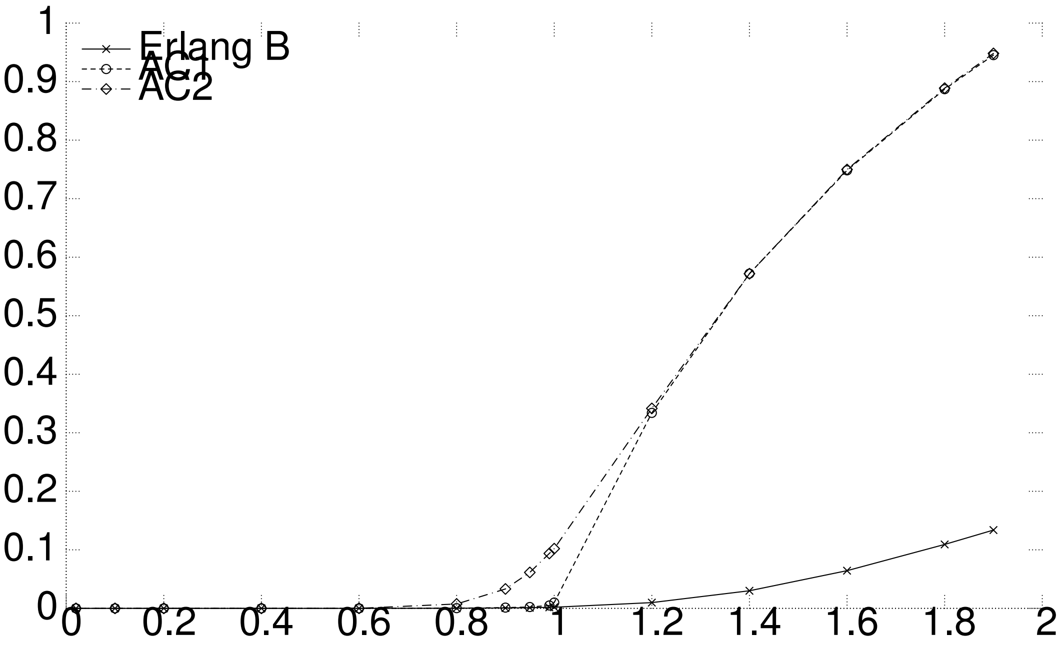
- Simulation of four models, with different arrival and service parameters
  - uncontrolled (denoted by UDP)
  - TCP-friendly (denoted by TCP)
  - adm control with $\varepsilon = \frac{1}{1000}$ (denoted by AC1)
  - adm control with $\varepsilon = \alpha$ (denoted by AC2)
- Based on simulation, we get the concept of how different schemes do to elastic and inelastic users

Typical probability density distribution:



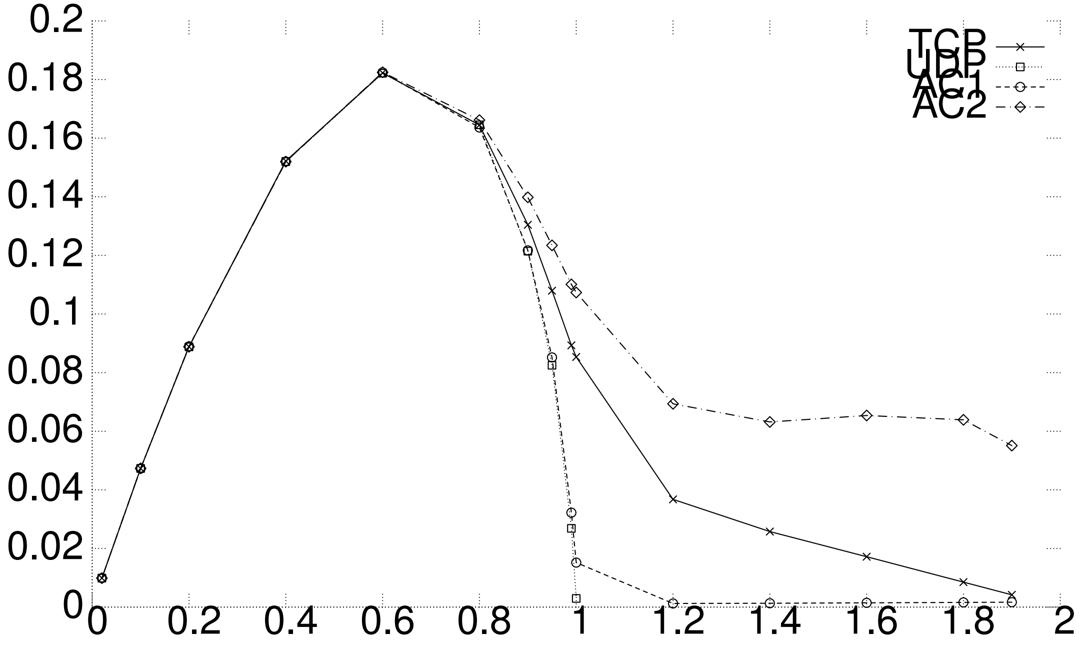
- The density is concentrated on small values

Blocking probability of admission control schemes:



- Compared with Erlang-B probability, the AC1 and AC2 give much higher blocking probability when $\rho > 1$

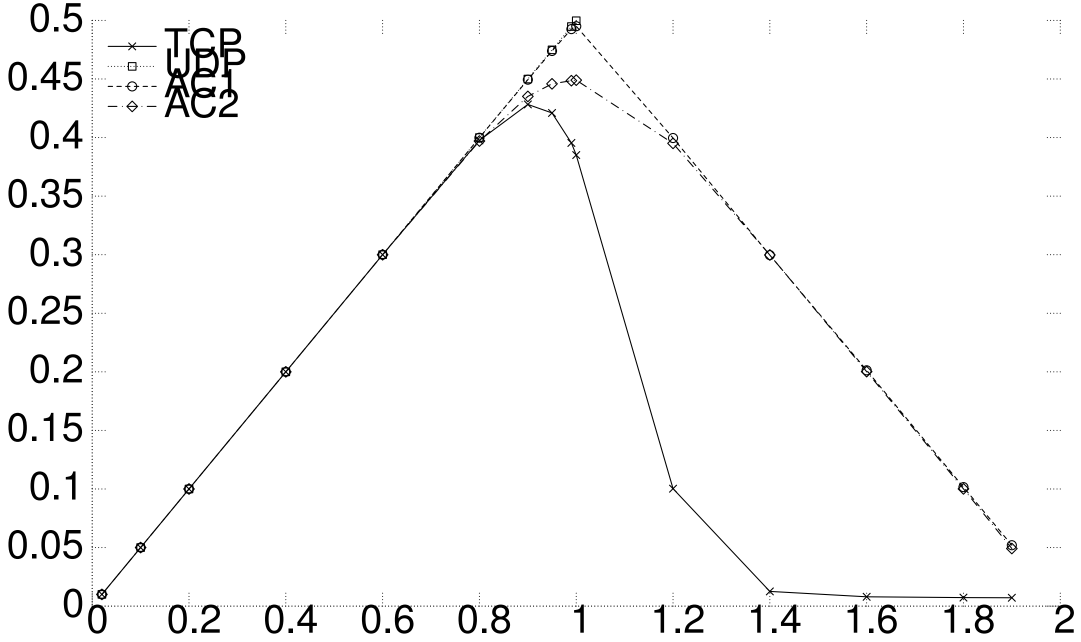
Elastic utility: (50% population is elastic)



- AC2 > TCP > AC1 > UDP

- UDP became unstable when $\rho_e + \alpha\rho_i > 1$

Inelastic utility: (50% population is inelastic)



- UDP > AC1 > AC2 > TCP
- UDP became unstable when $\rho_e + \alpha\rho_i > 1$

From the simulation, we can see that:

- Uncontrolled approach gives inelastic flow the best utility, but:
  - Stable region is only $\rho_e + \alpha\rho_i \leq 1$
  - Uncontrolled approach hurts elastic flow seriously
- TCP-friendly congestion control is good, but
  - It don't give enough utility to inelastic user, esp. when the network is congested

From the simulation, we can see that:

- AC2 (Admission Control with parameter ε set to α) is
  - Always better than TCP-friendly or uncontrolled when it is congested
  - Probably the way we should do for inelastic traffics
- Result from AC1 (with ε set to $\frac{1}{1000}$) shows that we can adjust ε according to how do we weigh the elastic flows over inelastic flows.

The road ahead:

- Use an analytical approach to tell the same story
- Performance bounds of different models, e.g. max supported population
- What is the appropriate value of ε for different scenarios?
- Any formula for blocking probability?
- Optimial λ and μ for optimizing the utility of a particular system?