A Measurement-Based Algorithm to Maximize the Utility of Wireless Networks

Julien Herzen

joint work with

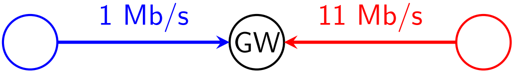
Adel Aziz, Ruben Merz, Seva Shneer and Patrick Thiran

September 19th, 2011

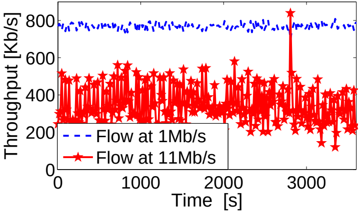
Context

Inefficient situations in wireless LANs

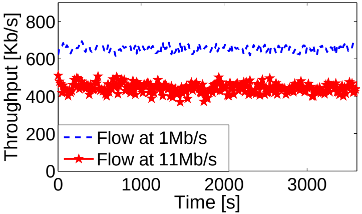
- **Example**, performance anomaly:



UDP

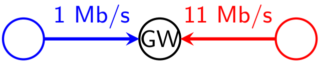


TCP



- **Intuition:** Send slightly fewer packets at 1 Mb/s, so that the flow at 11 Mb/s can send many more

Approach



- **Formalization:** Capture the efficiency and the fairness of the network using a utility function

$$U = \sum_i u_i(x_i),$$

x_i: throughput of flow i

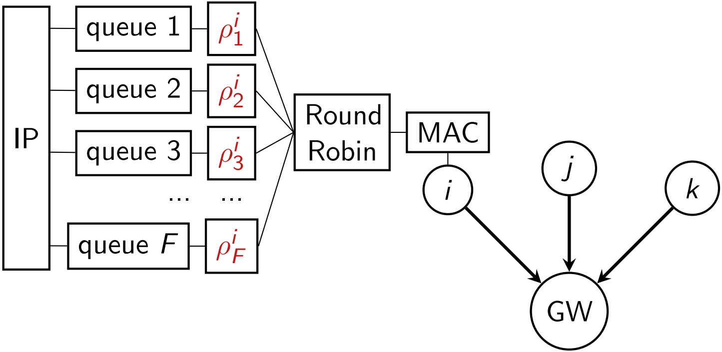
Examples

$$U_{max} = \sum_i x_i$$

$$U_{prop} = \sum_i \log x_i$$

Network Stack

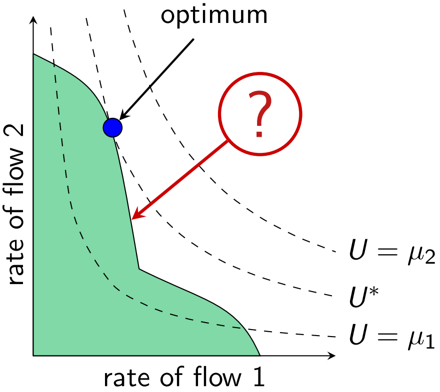
- Backward compatibility → runs on top of IEEE 802.11
- Congestion control → throttle each flow
- One limiter per IP source in the network



How to throttle the flows?

- Find the rate allocation $\boldsymbol{\rho}$ that maximizes the utility U
- **Problem: We do not know the feasible rate region!**
  - hard to predict or measure

$$\boxed{U = \sum \log x_i}$$



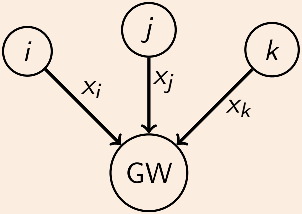
Decide at the gateway

The gateway knows

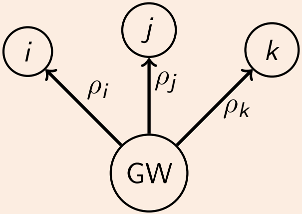
- The throughput achieved by the flows: $\mathbf{x}$
- The current utility of the network: $U(\mathbf{x}) = \sum u_i(x_i)$
- If $\mathbf{x} = \boldsymbol{\rho}$, then $\boldsymbol{\rho}$ belongs to the rate region

Measure and Decide loop

- Measure each flow
- Decide
- Broadcast decision



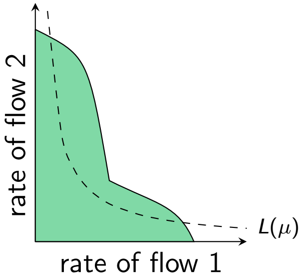
?



Model

At time slot n:

- Measured throughput: $\mathbf{x}[n] \in \mathbb{R}_+^F$
- Rate allocation vector: $\boldsymbol{\rho}[n] \in \mathbb{R}_+^F$
- Last stable rate allocation: $\mathbf{r}[n] \in \mathbb{R}_+^F$
- Utility function: $U(\mathbf{x}) = \sum_i u_i(x_i)$

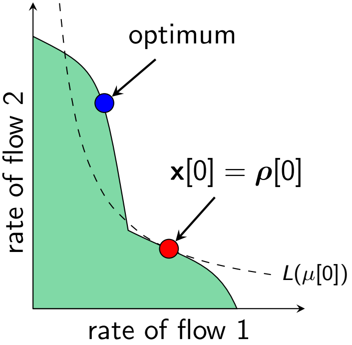


- Level set: $L(\mu[n]) = \{\mathbf{x}[n] : U(\mathbf{x}[n]) = \mu[n], \mathbf{x}[n] \in \mathbb{R}_+^F\}$

Step 1 - Start from IEEE 802.11 allocation

- $\boldsymbol{\rho}[0] \leftarrow \mathbf{x}[0]$
- Current level set: $L(U(\mathbf{x}[0]))$
- Remember allocation $\mathbf{r}[0] \leftarrow \mathbf{x}[0]$

$$\boxed{U = \sum \log x_i}$$

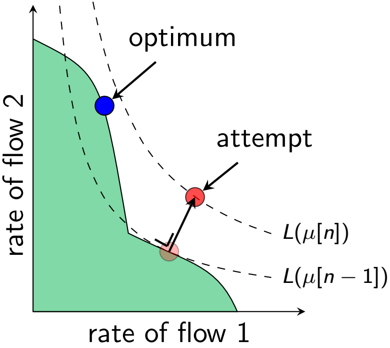


Step 2 - Enhance phase

Time step n:

- If $\mathbf{x}[n-1] = \boldsymbol{\rho}[n-1]$:
  - Obtain a new target utility $\mu[n]$ by a **full size gradient ascent**
- Else:
  - Obtain a new target utility $\mu[n]$ by **halving the size of the gradient ascent**
- Go to Explore phase (next slide)

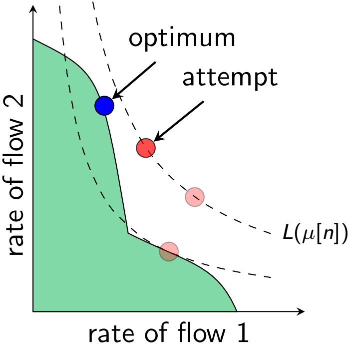
$$\boxed{U = \sum \log x_i}$$



Step 3 - Explore phase

- If $\mathbf{x}[n-1] = \boldsymbol{\rho}[n-1]$:
  - Remember $\mathbf{r}[n] = \boldsymbol{\rho}[n-1]$
  - Go to Enhance phase
- Else:
  - Keep target utility: $\mu[n] = \mu[n-1]$
  - Pick $\boldsymbol{\rho}[n]$ randomly in $L(\mu[n])$
  - Repeat explore phase at most N times, then move to Enhance phase (and **reduce the size of the gradient ascent**)

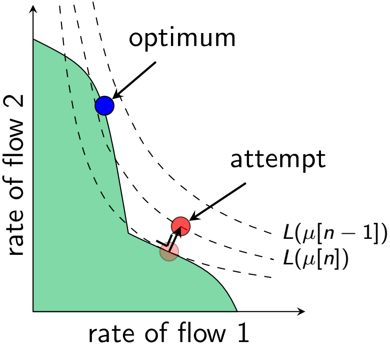
$$\boxed{U = \sum \log x_i}$$



Step 3 - Explore phase

- If $\mathbf{x}[n-1] = \boldsymbol{\rho}[n-1]$:
  - Remember $\mathbf{r}[n] = \boldsymbol{\rho}[n-1]$
  - Go to Enhance phase
- Else:
  - Keep target utility: $\mu[n] = \mu[n-1]$
  - Pick $\boldsymbol{\rho}[n]$ randomly in $L(\mu[n])$
  - Repeat explore phase at most N times, then move to Enhance phase (and **reduce the size of the gradient ascent**)

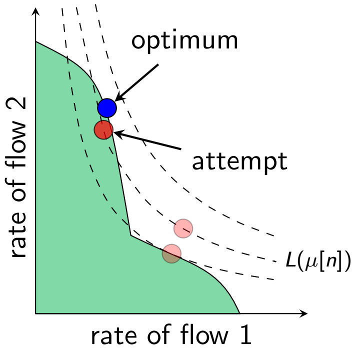
$$\boxed{U = \sum \log x_i}$$



Step 3 - Explore phase

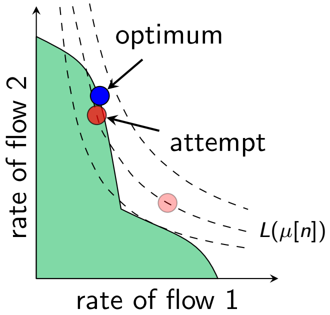
- If $\mathbf{x}[n-1] = \boldsymbol{\rho}[n-1]$:
  - Remember $\mathbf{r}[n] = \boldsymbol{\rho}[n-1]$
  - Go to Enhance phase
- Else:
  - Keep target utility: $\mu[n] = \mu[n-1]$
  - Pick $\boldsymbol{\rho}[n]$ randomly in $L(\mu[n])$
  - Repeat explore phase at most N times, then move to Enhance phase (and **reduce the size of the gradient ascent**)

$$\boxed{U = \sum \log x_i}$$

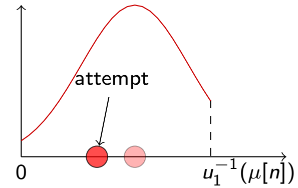


Step 3 - Explore phase

- If $\mathbf{x}[n-1] = \boldsymbol{\rho}[n-1]$:
  - Remember $\mathbf{r}[n] = \boldsymbol{\rho}[n-1]$
  - Go to Enhance phase
- Else:
  - Keep target utility: $\mu[n] = \mu[n-1]$
  - Pick $\boldsymbol{\rho}[n]$ randomly in $L(\mu[n])$
  - Repeat explore phase at most N times, then move to Enhance phase (and **reduce the size of the gradient ascent**)



truncated Gaussian PDF for picking ρ_1:



Optimality result

Assumptions

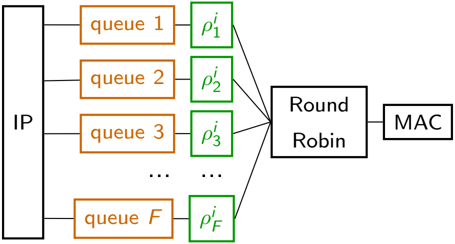
- Fixed rate region $\Lambda[n] = \Lambda$
- Coordinate-convex rate region
    - **Much weaker than convexity!**

Theorem

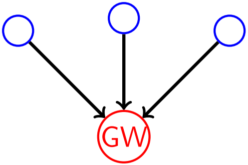
The Enhance & Explore algorithm guarantees that, for any initial rate allocation $\mathbf{r}[0]$*, the utility of the last stable rate allocation* $\mathbf{r}[n]$ *converges to the maximal utility for* $n \to \infty$*.*

Practical implementation

- Based on *Click*[1] with *MultiflowDispatcher*[2]



- Creation of 4 new *Click* elements
  - MFQueue
  - MFLeakyBucket
  - EEadapter
  - EEscheduler
- Evaluation with
  - Asus routers
  - *ns-3*

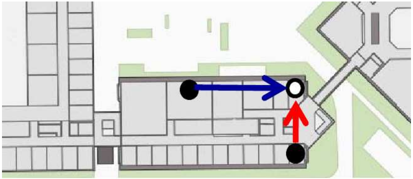


[1] Kohler et al., Transactions on Computer Systems, 2000

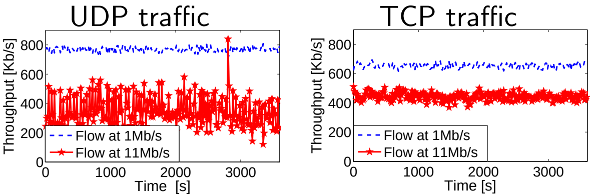
[2] Schiöberg et al., SyClick, 2009

Experimental results

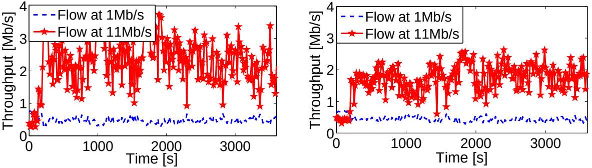
- Deployment map:



- Without E&E:

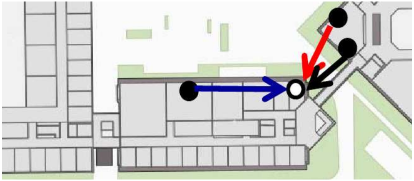


- With E&E (U_{prop}):

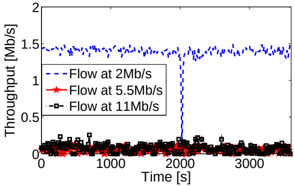


Experimental results

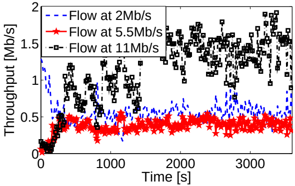
- Deployment map:



- Without E&E:



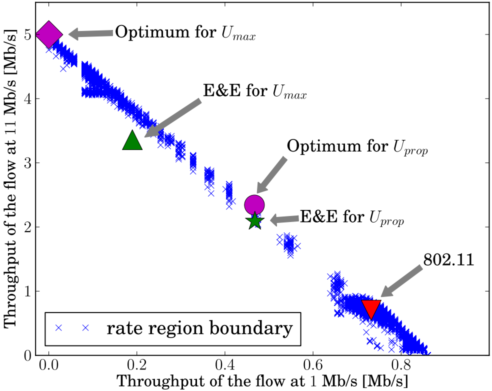
- With E&E (U_{prop}):

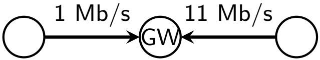


Simulation results

ns-3 simulator

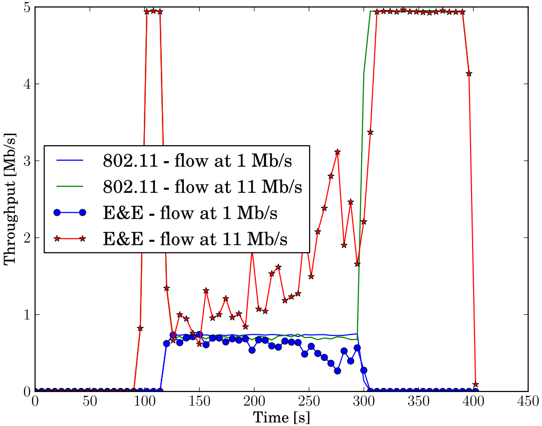
- Re-use of the same *Click* elements
- More controlled environment
- Possible estimation of the rate region
- Computation of optima





Simulation results

- Adaptivity to time-varying traffic
- Cyclic validation of last stable allocation $\mathbf{r}[n]$



Conclusion

Problem

- Inefficient and/or unfair situations in WLANs
- Capture efficiency and fairness using a utilility function
  - **The feasible rate region is unknown!**

Solution

- Successive decisions and measurements by the GW
- Optimal for a fixed rate region
- When rate region changes, keeps adapting
- More details in [1], with an extension to multi-hop networks

Future work:

- Downlink traffic
- Rate adaptation

[1] Aziz et al., Mobicom 2011