

# Removing the MAC Retransmission Times from the RTT in TCP

**E. Dedu, S. Linck, F. Spies**

Laboratoire d'Informatique de l'Université de  
Franche-Comté (LIFC)  
Montbéliard, France

Euromedia'2005  
Toulouse, France  
11 April 2005

# Problem: RTT modification

- TCP works very well in wired links
  - very few physical losses
  - most losses are due to network congestion
    - TCP reduces packet rate, in order to eliminate congestion
- TCP is not adapted to wireless links  $\Leftarrow$  interferences
  - short  $\Rightarrow$  MAC retransmissions  $\Rightarrow$  increase of RTT
    - RTT as congestion indicator (queue length) is no longer appropriate
    - RTO depends on RTT, it is falsely modified too
  - long  $\Rightarrow$  lost packets
    - $\Rightarrow$  inappropriate congestion control actions, since not congestion



# Problem: RTT effects

- Several CC mechanisms use the RTT:
  - each RTT: TCP Vegas
  - the smallest RTT: Westwood+, TIBET
  - any solution where RTT might be used: RTP/RTCP over UDP
- TCP Vegas: for each packet reception, it compares its RTT against the estimated RTT:
  - if  $\text{diff} < \alpha$ , reduces rate
  - if  $\alpha < \text{diff} < \beta$ , maintain rate
  - if  $\text{diff} > \beta$ , increases rate

# Proposition

- Put in packets the time spent on MAC retransmissions
- A TCP option is added
- Network cards have a timer
- Cards add to the TCP option the time lost in MAC retransmissions
- Receiver echoes back the time
- Source takes the appropriate action

# Plan

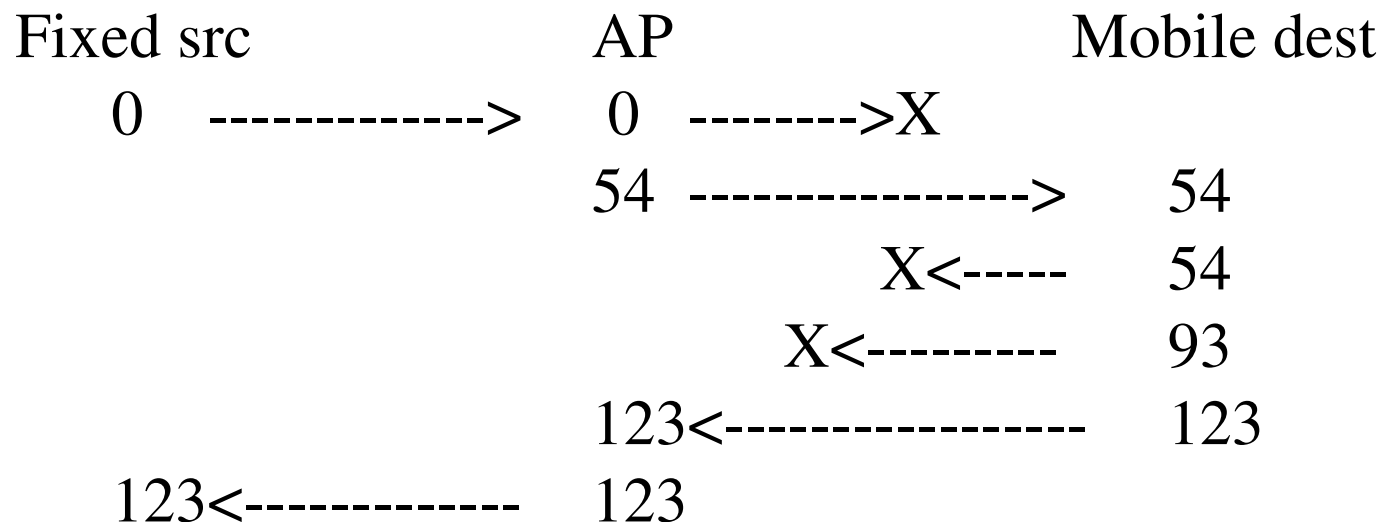
- Principle
  - transmitting information to the source
  - retransmission time computing
- Simulations
  - ns2 modification
  - background: shadowing wireless propagation model
  - results
- Related work
- Conclusions, future work

# Principle: transmitting information

- TCP option:
  - unit:  $20\mu s$
  - size: 2 bytes  $\Rightarrow 65536 * 20\mu s \simeq 1.3s$ 
    - 802.11b standard: max 1023 or retransmission window
- Source sets the field to 0
- The field is modified by network cards
- Receiver echoes back the value of the corresponding data packet (exactly like timestamp option)
- Source receives the information
- $\Rightarrow$  Incremental deploying possible

# Principle: retransmission time computing

- Each network card has a timer
- For each packet:
  - when the packet is sent, the timer is initialised with the value of the TCP option
  - each time the packet is resent, the value of the timer is stored in the option => lost times are added



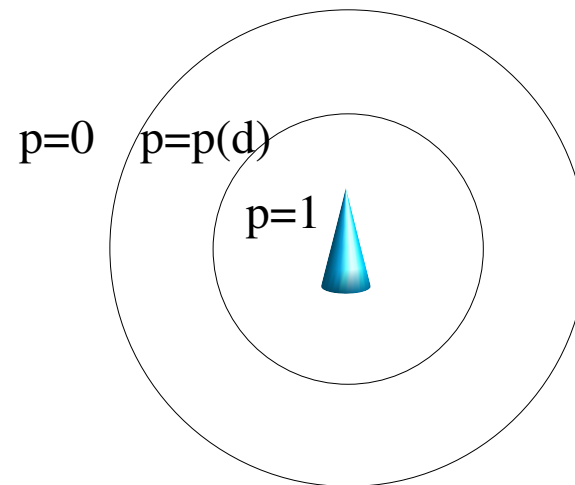
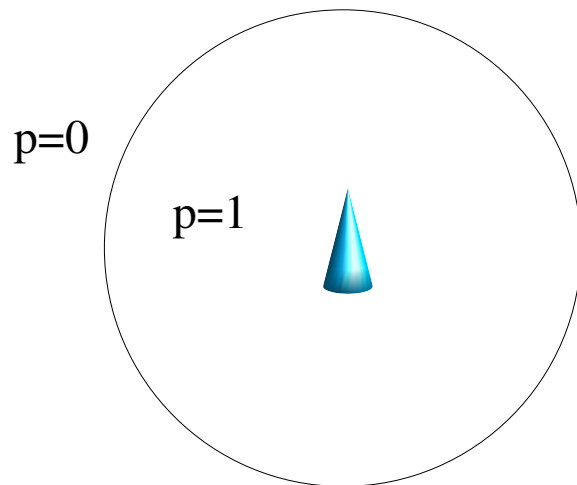
# Simulations: ns2 modifications

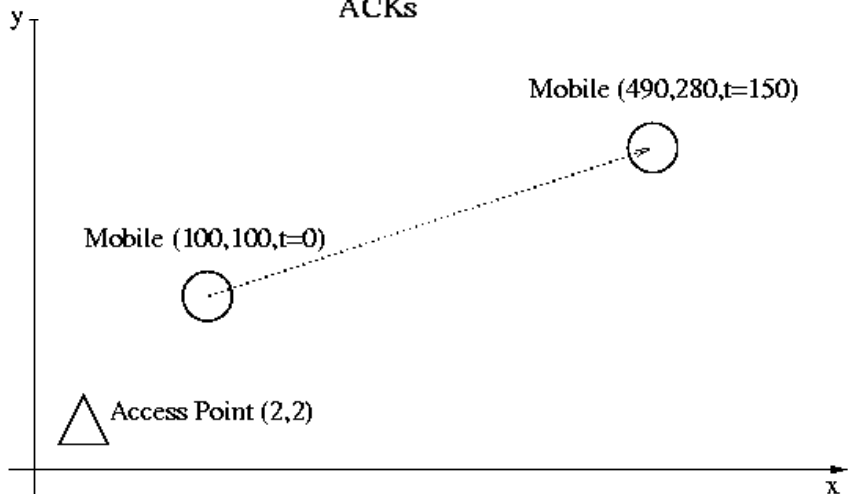
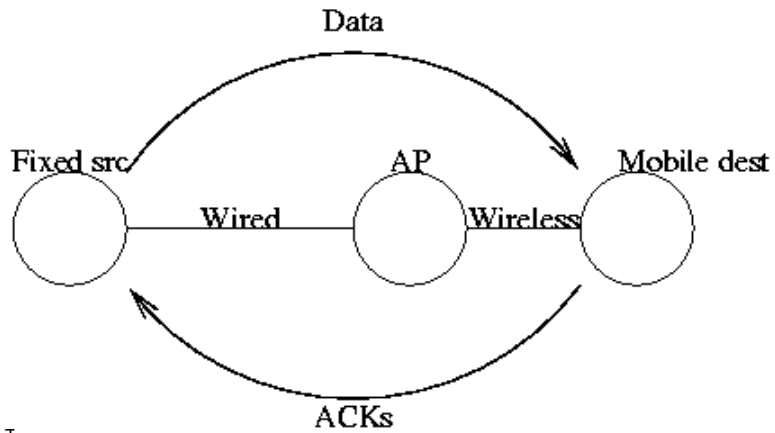
- Addition of the rets field to TCP header
- The sending part of TCP/Vegas
- The sending part of 802.11
  - value used: the time between packet sending time and the reception of its MAC ACK
- Sending part of TCPSink
- Receiving part of TCP/Vegas
  
- All tests are available on Web page



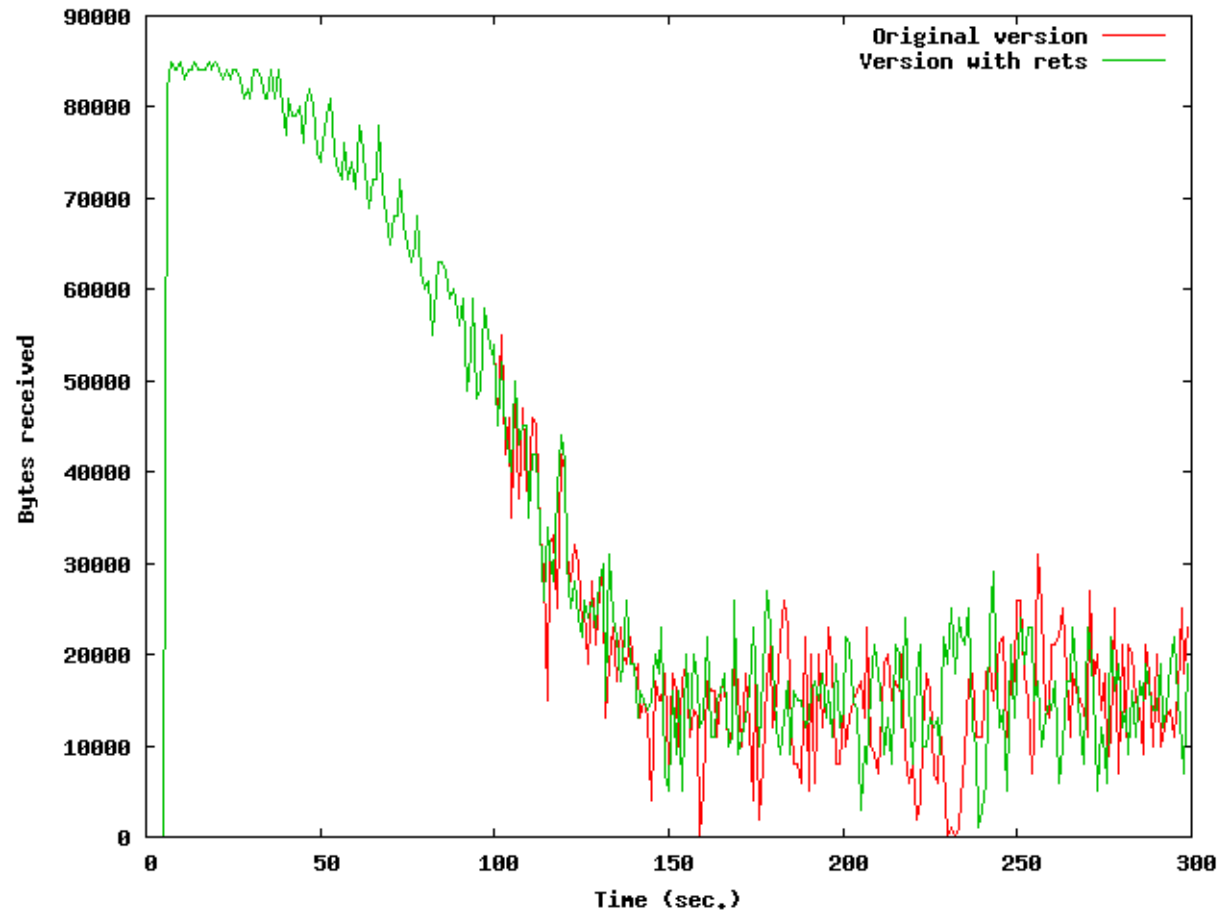
# Background: propagation models in ns2

- All or nothing:
  - FreeSpace
  - TwoRayGround
- Probability-based after a certain distance
  - Shadowing  $\leq$  used in the tests





# Simulations: results



max 1Mb/s

- $t < 5$ , ftp starts at 5s
- $5 < t < 100$ , identical
- $150 < t$ , 4.5% improvement
- $150 < t < 250$ , 15% improvement

# Related work: general

- Packet loss in wireless:
  - [Jing et al. 2000], a timestamp is added by AP in each packet  
=> can detect packet loss even for out-of-order packets =>  
source is informed faster about packet loss
- RTT influence on RTO:
  - [Scharf et al. 2004], analytical model of RTO based on network parameters
  - [Möller et al. 2004], artificial delay at AP => RTT increases  
=> RTO increases => false TCP retransmissions decreases =>  
higher overall throughput

# Related work: packet delay in wireless

- [Ratnam et al. 1998], AP divides the connection in two connections and estimates the packet delay
- Integrates within timestamp TCP option
  - timestamp granularity is machine-dependent
- => Each time a packet is resent, its timestamp is replaced with the most recent timestamp of the same flow
  - => estimation
  - => network cards: memory and CPU consuming
- Results: the more losses in wired part of connection, the higher the throughput

# Conclusions

- A method to remove the effect of packet MAC delay
- Network cards have a timer which is added to the TCP option of the packet
- Receiver echoes back the value of the option
- Simulations: improvement of bandwidth

# Drawbacks and future work

- Network cards have access to level 4
- Deployment: source and destination (and AP)
- Restricted use (uses RTT)
  
- Analysis of more complex scenarios
  - competing flows in wireless
  - lossy wired links
- Real experiments (pb.: access the MAC firmware)
- If successful, define a complete proposal