# MulTFRC:
# TFRC with weighted fairness

*draft-irtf-iccrg-multfrc-00*

Michael Welzl, Dragana Damjanovic,
Stein Gjessing

ICCRG @ 78th IETF Meeting
Maastricht, Netherlands
30 July 2010

# What is MulTFRC?

- Like MulTCP: a protocol that is *N*-TCP-friendly
  - $N \in R^+$
  - Larger range of possible values for *N* than for others, e.g. MulTCP and CP
  - Yields flexible weighted fairness (e.g. priorities between users, or between flows of a single user)

- Based on TFRC
  - Easy to implement as an extension of TFRC code
  - Change the equation + measure “real” packet loss

# Research background

- Ph.D. thesis of Dragana Damjanovic
  (now finished and evaluated with best marks)
  - Equation derivation: SIGCOMM poster, tech. rep., paper with derivation + MulTFRC under submission
  - MulTFRC: CCR paper
- Extensive evaluations: equation validation, MulTFRC tests, both in simulations and real life
  - MulTFRC also successfully demonstrated for Europe-China file transfer at final review of European IST FP6 STREP project "EC-GIN"
- All documentation and code available from:
  http://heim.ifi.uio.no/michawe/research/projects/multfrc/

# Draft history

- draft-welzl-multfrc-00 presented at ICCRG meeting, IETF 75 in Stockholm
  - General feedback positive (but not much feedback)
- draft-welzl-multfrc-01 presented at DCCP meeting, IETF 76 in Hiroshima
  - General feedback positive (but not much feedback)
  - Decision: this is more appropriate for ICCRG
- ICCRG reviews: Wes Eddy, Lachlan Andrew, Dirceu Cavendish
  - Main criticism: algorithm looks complicated, might be a problem to implement (overflows etc.)
  - Addressed in this update. Dirceu already said OK

# Overview of changes

1. Syntactic change in the algorithm that computes the sending rate, X_Bps

2. Special treatment of the case that the loss event rate is one

3. Arguing that there will be no underflow, overflow or rounding errors in the algorithm that calculates X_Bps

# Section 2.1: Change in X_Bps

"Syntactic" change in the algorithm that computes the sending rate, X_Bps:

```
If (q*z/(x*R) >= N) {
    q = N;
 } Else {
    q = q*z/(x*R);
 }
```

changed to:

```
q = min(q*z/(x*R),N);
```

# Section 2.2: p==1

When all packets are lost (the loss event rate, p, is one) the algorithm for computing X_Bps will perform a division by 0. The case p==1 is now treated as a special case before the algorithm is invoked:

The procedure for updating the allowed sending rate in section 4.3 of [RFC5348] ("action 4") contains the statement:

Calculate X_Bps using the TCP throughput equation.

which is replaced with the statement:

```
If (p==1) {
   X_Bps=s*N/t_mbi;
} Else {
   Calculate X_Bps using the algorithm defined in section 3.
}
```

*Note, small mistake:*
*N is missing in the current draft. fixed in the next update.*

s: Nominal packet size in bytes.

t_mbi: Maximum RTO value of TCP (constant)

# Appendix: The calculation of X_Bps

- Show that when p != 0 and p != 1,
  our algorithm for calculating X_Bps
  does not give underflow, overflow or rounding errors.
- The appendix goes through all calculations
  in the algorithm and shows that all operations
  have valid operands and produce valid results
  (when the other parameters also have proper values (like s, R, b, ...)).

# Next steps

- Going to submit a minimal update immediately after this meeting
  - Would like to submit this for IRSG review soon afterwards
  - Please provide feedback fast

- Planned future work: DCCP CCID specification, maybe also a small-packet variant