GridDT: TCP pentru retele de mare viteza

Coordonator stiintific:
Prof. Dr. Ing. Nicoale TAPUS

Consultant:
Sylvain Ravot, CERN

Absolvent:
Cristian ORBAN

Scopul proiectului

- GridDT (Grid Data Transport) este o versiune imbunatatita de TCP (Transmission Control Protocol) pentru retele cu produs bandwidth-delay mare
- Dezvoltarea GridDT a inceput in 2002
- Implementarea GridDT consta intr-un patch pentru kernel-ul Linux
- Scopul acestui proiect este dezvoltarea unei versiuni GridDT imbunatatite (versiunea 3.1)

Cum functioneaza TCP?

- TCP este un protocol care se incadreaza la nivelul transport al modelului OSI
- TCP asigura transmiterea sigura
- TCP foloseste un mecanism de tipul fereastra glisanta (sliding window)
  - Fereastra reprezinta cantitatea de date ce se pot injecta in retea de catre emitator fara sa astepte confirmare
- TCP este orientat pe conexiune → exista un mecanism pentru stabilirea unei conexiuni intre emitator si receptor inainte de a transmite
- O conexiune TCP trece prin 3 faze:
  - Stabilirea conexiunii
  - Transmisia propriu-zisa
  - Incheierea conexiunii

Cum functioneaza TCP?

- Transmisia propriu-zisa este controlata de 2 mecanisme TCP: slow-start si congestion avoidance (evitarea congestiei)
  - Legate de aceste mecanisme sunt variabilele TCP cwnd (congestion window) si ssthresh (slow start threshold)
  - Algoritmul folosit de TCP este urmatorul:
    - cat timp cnwd <= ssthresh, se face slow start, si cwnd este incrementat cu 1 pentru fiecare confirmare primita
    - dupa ce cwnd > ssthresh, se trece la evitarea congestie, iar cwnd este incrementat cu 1/cwnd pentru fiecare confirmare primita
    - In cazul in care nu se primeste o confirmare pentru un segment si timer-ul pentru retransmiterea segmentului respectiv expira, dupa retransmisie cwnd devine 1 si se face slow-start

De ce este GridDT necesar?

- Exista din ce in ce mai multe aplicatii care necesita performante sporite in ceea ce priveste transferul peste retele cu produs bandwidth-delay mare
- De asemenea capacitatile legaturilor avanseaza rapid, facand absolut necesara nevoia pentru un protocol ce poate transfera mai mult de 1Gbps
- De ce este TCP ineficient?
  - S-a dovedit ca atunci cand latimea de banda si intarzierea pe o anumita legatura cresc, TCP devine instabil
  - Algoritmul AIMD (Additive Increase Multiplicative Decrease) folosit de TCP este ineficient in retele cu produs bandwidth-delay mare: incrementarea cwnd cu 1/cwnd in timpul evitarii congestiei este mult prea conservativa, iar injumatatirea cwnd cand apare congestie prea drastica
  - O alta potentiala problema este agresivitatea algoritmului Slow Start
  - De asemenea, modul in care se face incrementarea cwnd nu este echitabil in cazul in care avem flow-uri cu parametri diferiti

Alte variante TCP imbunatatire

- Exista mai multe variante de TCP imbunatatit; FAST este una dintre acestea
- FAST este dezvoltat la Caltech
- FAST foloseste o abordare inovativa in detectia congestiei: se bazeaza atat pe pierderi cat si pe intarziere
- GridDT va fi evaluat comparativ cu FAST in cadrul acestei lucrari

Proiectarea GridDT

- Scopurile procesului de design au fost urmatoarele:
  - Obtinerea unei rate de transfer cat mai bune
  - Revenire dupa pierderi intr-un timp cat mai scurt
  - Modificari decat la partea de emitator
  - Nu va necesita suport din partea ruterelor
  - Nu va necesita suport din partea receptorului

Implementarea GridDT 3.1

- Urmatoarele modificari sunt prezente in GridDT 3.1:
  - Algoritmul AIMD a fost modificat: incrementul utilizat in perioada de evitare a congestiei este calculat in functie de parametri flow-ului (MTU, RTT); de asemenea, la aparitia congestiei cwnd nu mai este injumatatit, ci decrementat dupa formula cwnd=cwnd-cwnd/x, unde x este parametru sysctl configurabil
  - Aceasta modificare a algoritmului AIMD poate duce la situatii cand cwnd este incrementat cu o valoare mare; pentru a evita astfel de situatii a fost implementat un mecanism care imparte aceasta crestere pe durata unui RTT

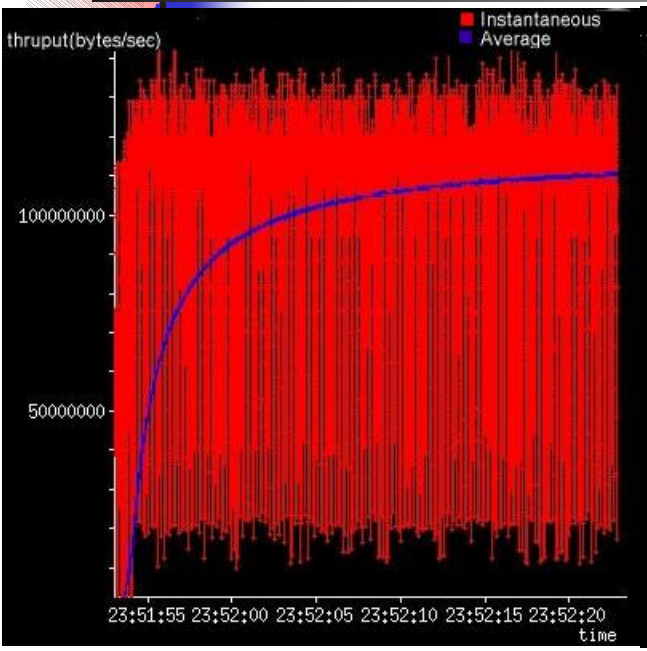
Implementarea GridDT

- A fost implementat algoritmul Limited Slow Start, propus in RFC 3742.
  - Acesta defineste o noua variabila, max_ssthresh, si schimba algoritmul Slow Start dupa cum urmeaza:
    - Cat timp cwnd < max_ssthresh, se face slow-start
    - Cat timp max_ssthresh<=cwnd<ssthresh se face Limited Slow Start, iar cwnd se incrementeaza cu 1/k, unde k=cnwd/max_ssthresh*0.5
- O alta imbunatatire propusa a fost implementarea DCR (Delayed Congestion Response). In mod normal, cand se primesc 3 confirmari duplicate, se face retransmisia segmentului respectiv (fast retransmit), iar pentru incrementarea cwnd se foloseste algoritmul din faza de evitarea a congestiei(fast recovery). DCR propune intarzierea raspunsului la congestie(retransmisiei), deoarece aceasta limita de 3 confirmari duplicate nu mai corespunde retelelor actuale.

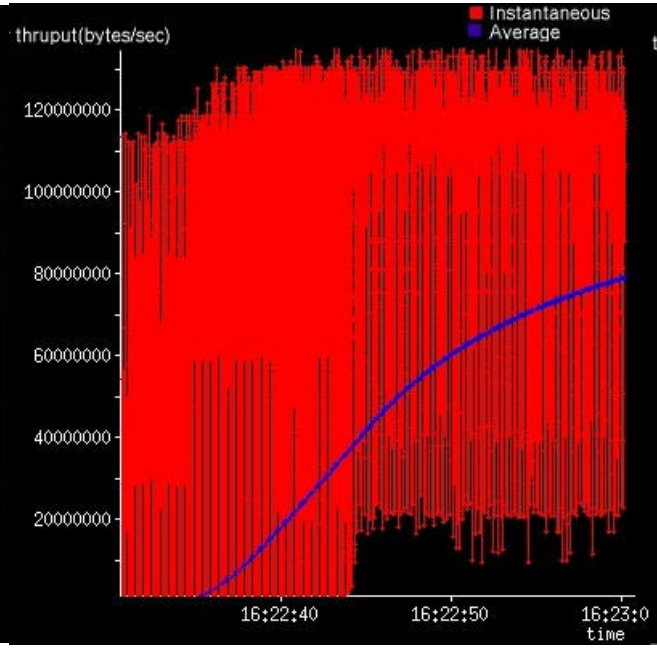
Testarea si evaluarea GridDT

- Testele s-au efectual utilizand reteaua Datatag
  - O legatura de 2.5Gbps intre Geneva si Chicago
  - Statii cu legaturi de 1Gbps
- Aplicatii utilizate:
  - Iperf – generator de trafic
  - Tcpdump – captura trafic
  - Tcptrace, xplot – grafice
  - Dummynet – pentru testarea mecanismului DCR
- 3 tipuri de teste:
  - Teste de transfer
  - Teste cu flow-uri concurente
  - Teste cu pierderi de pachete

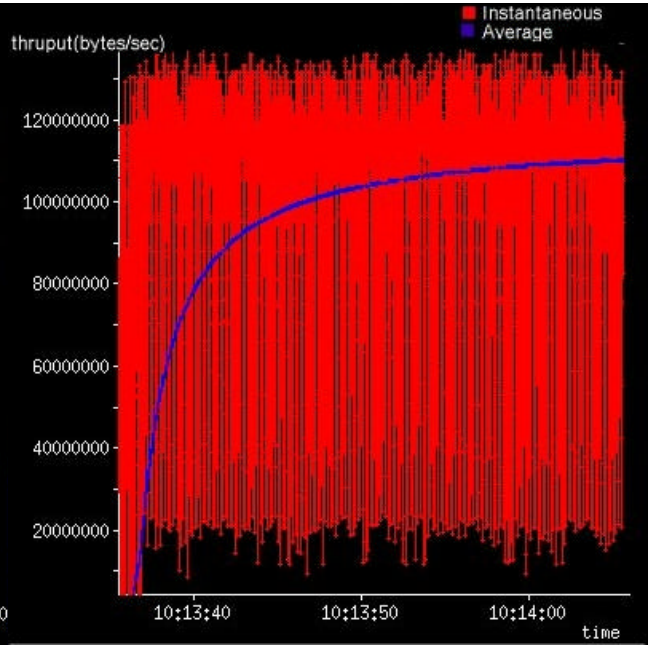
Teste de transfer



TCP, 893Mbps



FAST, 787Mbps

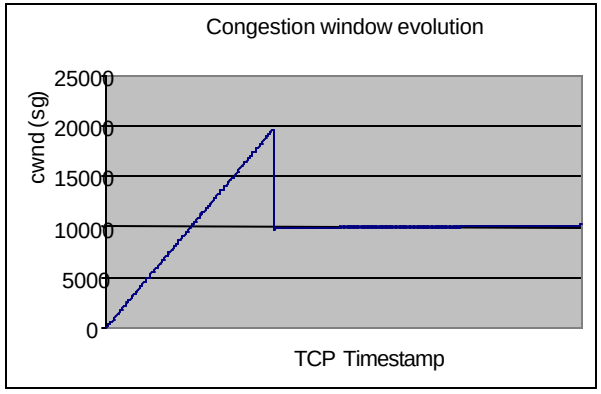


GridDT, 894Mbps

- Testul pentru GridDT nu s-a facut cu max_ssthresh=100, asa cum se propune in RFC3742 pentru Limited Slow Start; valoarea aleasa pentru aceasta legatura a fost 8000

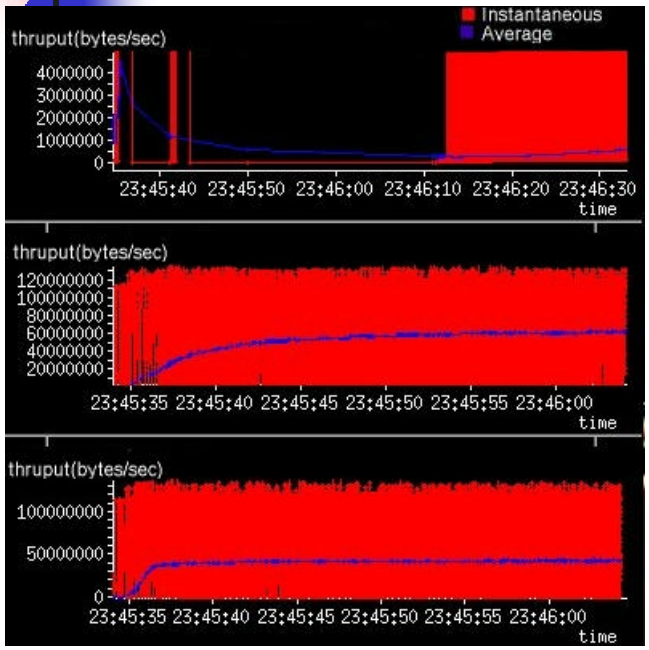
Teste de transfer

- Determinarea valorii optime pentru max_ssthresh a avut in vedere urmatoarele deziderate:
  - obtinerea unei cresteri relativ rapide al latimii de banda utilizate
  - o crestere mult mai putin agresiva decat in slow-start dupa ce se depaseste max_ssthresh
- Pentru aceasta determinare s-a trasat graficul de mai jos, care prezinta evolutia cwnd pe parcursul unui test de 30s, cu max_ssthresh infinit

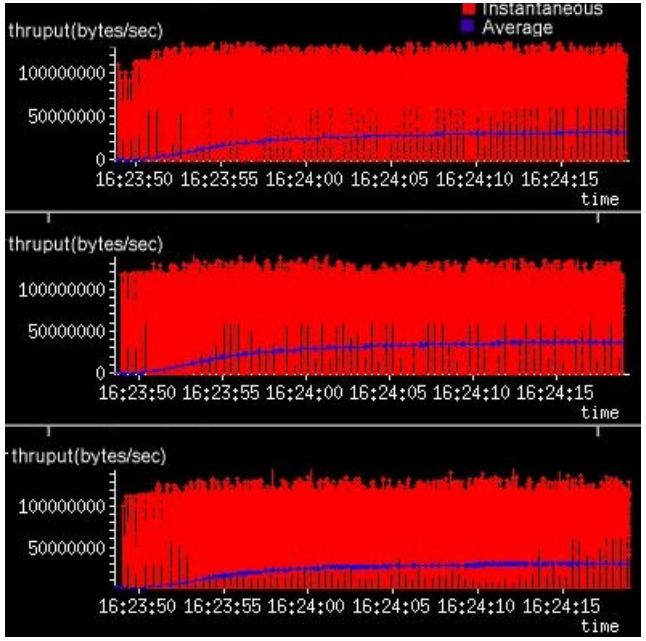


- Din acest grafic se observa aparitia congestiei in jurul valorii de 20.000 de segmente pentru cwnd; rezulta astfel o valoare de 10.000 segmente pentr cwnd si ssthresh dupa tratarea congestiei
- Valoarea de 8000 pentru max_ssthresh a fost aleasa in ideea asigurarii unei cresteri a cwnd dupa ce s-a depasit max_ssthresh cu ½ mai putin agresiva decat in slow-start

Teste cu flow-uri concurente



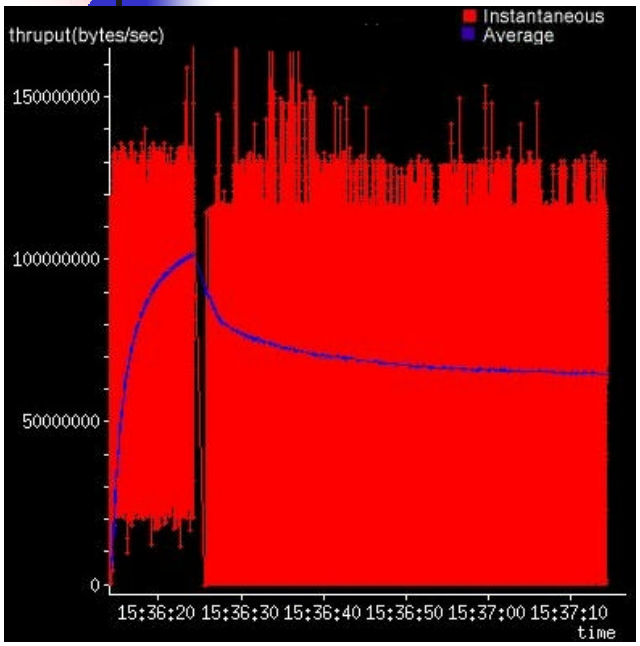
TCP



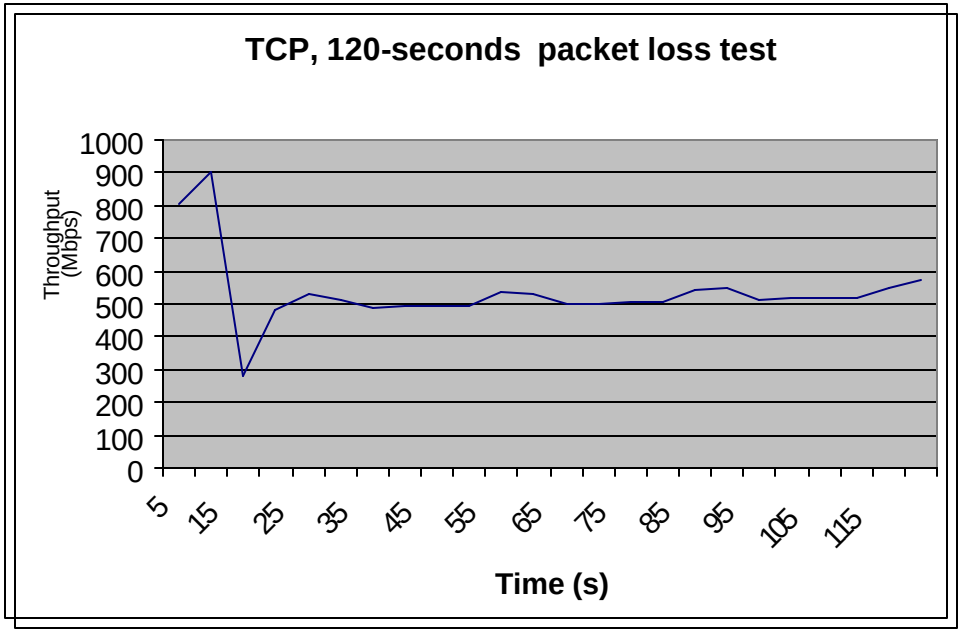
FAST

- Este evident comportamentul nesatisfacator al TCP: rata de transfer a primului flow scade dupa ce apar si celelalte
- FAST s-a comportat excelent
- GridDT a fost de asemenea testat, dar rezultatele au fost doar putin mai bune decat ale TCP

TCP: Teste cu pierderi de pachete



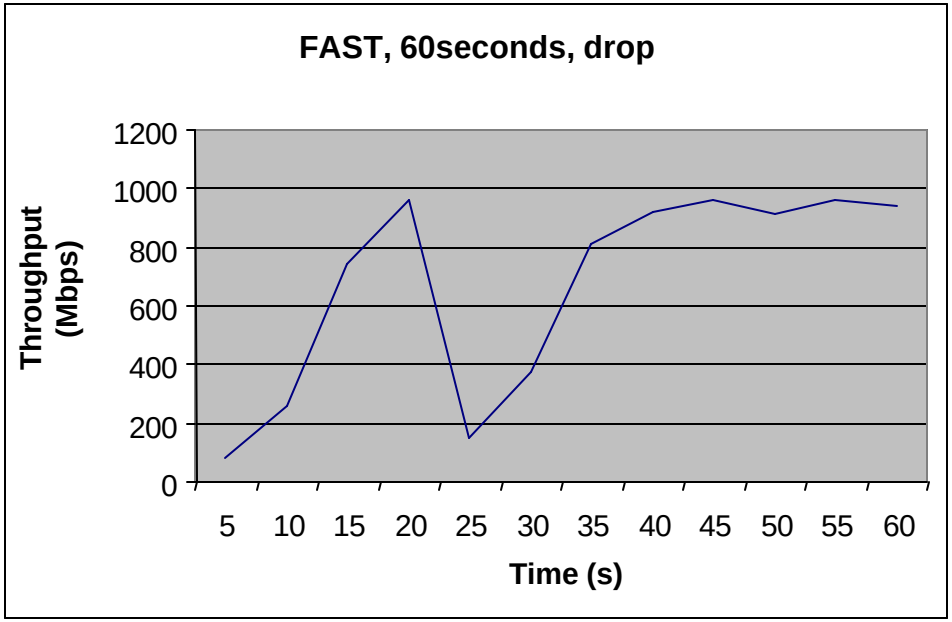
TCP, 30s



TCP, 120s

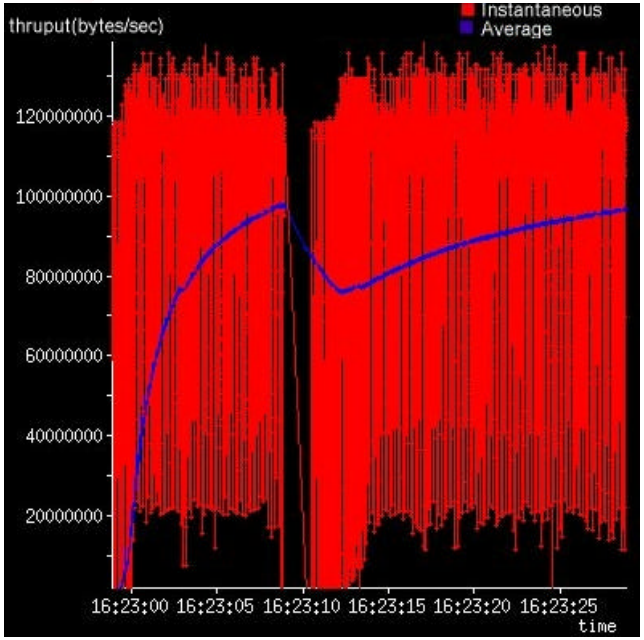
- Pierdere de pachete a fost simulata utilizand iptables pentru a filtra pachete pentru 0.5s pe parcursul testului de 30s, dupa ce a fost atinsa utilizarea legaturii > 90%
- Rezultatele TCP sunt nesatisfacatoare: chiar dupa 120s, TCP nu mai revine la nivelul de utilizarea al legaturii anterior pierderii

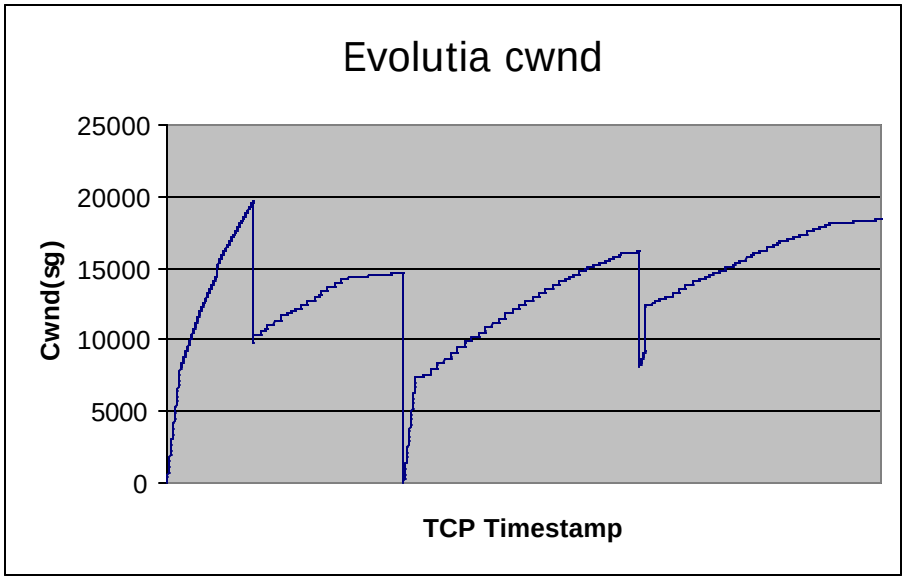
FAST: Teste cu pierderi



- testul FAST a durat 60s (deoarece a fost nevoie de mai mult timp pentru a atinge utilizarea peste 90% a legaturii)
- Simularea de pierdere de pachete a durat 0.5s
- FAST s-a comportat foarte bine, fiind necesarea aproximativ 20s pentru revenirea la o utilizarea a legaturii > 90%

GridDT: Teste cu pierderi





GridDT, 30s

- simularea de pierdere de pachete a durat 0.5s
- GridDT s-a comportat excelent, timpul de revenire la o utilizare a legaturii de peste 90% a fost ~10s
- Graficul din dreapta prezinta evolutia cwnd pe parcursul testului. Este evident faptul ca au avut loc pierderi de pachete, deoarece la un moment da cwnd=1

Testarea DCR

- Testarea mecanismului DCR s-a realizat utilizand dummynet
- Dummynet este o aplicatie FreeBSD, special proiectata pentru testarea protocoalelor
- Dummynet se foloseste pe un ruter/bridge ruland FreeBSD pentru:
  - simulare intarzierilor
  - Simularea unor legaturi cu anumita latime de banda
  - Simularea efectului de cai multiple
  - Simularea intarzierii datorata cozilor de pe interfetele ruterelor
- **In cazul GridDT, dummynet a fost folosita pentru a simula efectul de cai multiple, generand astfel reordonarea pachetelor la receptor**
  - Testarea s-a realizat folosind 2 statii Linux si un bridge cu FreeBSD si dummynet
  - Pachetele apartinand sesiunii dintre cele 2 masini de test au fost impartite in 3 categorii, iar pachetelor din fiecare categorie li s-a aplicat o intarziere diferita
- S-a constatat ca DCR aduce o imbunatatire minora, castigul in cazul testat fiind de aproximativ 3Mbps

Concluzii

- TCP standard este nepotrivit pentru retele cu produs bandwidth-delay mare
- FAST aduce imbunatatiri semnificative. FAST a obtinut rezultate foarte bune la toate tipurile de teste prezentate anterior. Dintre protocoalele testate a obtinut cele mai bune rezultate la testul cu flow-uri concurente
- GridDT a obtinut cele mai bune rezultate la testele cu pierderi de pachete. Au fost obtinute rezultate foarte bune si la testele de transfer.
- Comportamentul GridDT atunci cand exista mai multe flow-uri in competitie pentru utilizarea latimii de banda nu este decat putin imbunatatit fata de cel al TCP. Eforturile viitoare pentru imbunatatirea GridDT trebuie indreptate in aceasta directie
- Mecanismul Delayed Congestion Response a aratat doar o imbunatatire minora. Teste efectuate in absenta fenomenului de reordonare au aratat ca acesta nu afecteaza negativ performanta