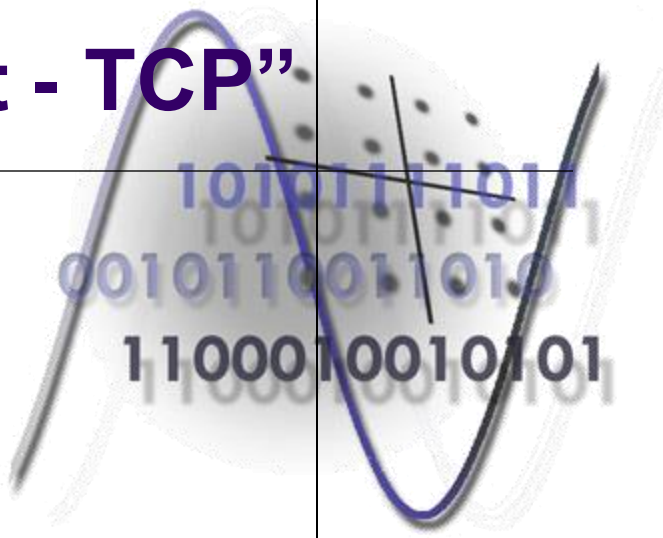
FEC-TCP - “Loss Tolerant - TCP”

Zsolt Polgar

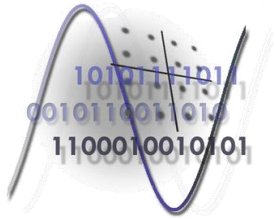
Communications Department
Faculty of Electronics and Telecommunications,
Technical University of Cluj-Napoca



Conţinutul prezentării

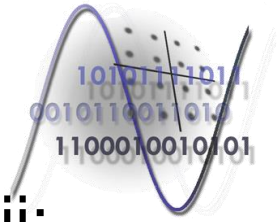
- Protocolul TCP – operaţii de bază;
- Protocolul TCP NewReno – operaţii de bază;
- Protocolul FEC-TCP bazat pe coduri DF;
- Modelul de canal utilizat;
- Evaluarea protocolului FEC-TCP;

Protocolul TCP – operaţii de bază



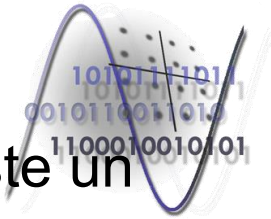
- Receptorul TCP realizează următoarele operaţii:
  - Acceptă pachete care nu respectă secvenţa corectă;
  - Bufferează aceste pachete şi le reordonează;
  - Declară o fereastră de recepţie W_{max}; se asigură că sursa nu transmite mai multe pachete decât această fereastră;
- Se consideră că aplicaţia poate accepta pachetele în momentul în care aceastea sunt disponibile;
- Receptorul returnează un *ack* pentru fiecare pachet corect;
- *Ack* sunt cumulative: *ack* cu nr. de secvenţă n validează toate pachetele până la inclusiv n-1;
  - Dacă se pierde un pachet, n, se trimte *ack* cu numărul lui de secvenţă, adică n;

Protocolul TCP – operaţii de bază



- Transmiţătorul TCP realizează următoarele operaţii:
  - La fiecare moment de timp transmiţătorul reţine următoarele variabile:
    - A(t) – limita inferioară a ferestrei; toate pachetele numerotate până la A(t)-1 sunt transmise şi validate;
      - Recepţia unui *ack* cu numărul de secvenţă n>A(t) face ca A(t) să sară la n;
    - W(t) – fereastra de congestie; transmiţătorul poate trimite pachete cu numărul de secvenţă n, A(t)<n<A(t)+W(t);W(t)<Wmax;
      - W(t) se modifică după o anumită regulă care permite modificarea debitului;
    - $W_{th}(t)$ – *"slow start threshold window"* ; controlează pasul de modificare al W(t);
- "Time-out" retransmisie
  - Pe baza măsurării întârzierii dintre pachetele transmise şi *ack* repepţionate, sursa poate calcula valoarea RTT ("Round Trip Time");

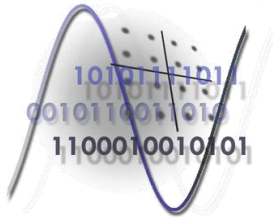
Protocolul TCP – operaţii de bază



- Pentru fiecare pachet nou transmis transmiţătorul porneşte un timer şi se resetează timerul care deja rulează (dacă un asemenea timer rulează deja);
- Timerul se setează la valoarea RTO (“Retransmission Time-Out”), valoare care se calculează din RTT;

- Adaptarea ferestrei şi recuperare pe bază de “time-out”;
  - Definiţie: Un pachet *ack* care validează prima recepţie a unui pachet corect şi aflat în secvenţa corectă – “*first ack*”;
  - Evoluţia normală a proceselor A(t), W(t) şi $W_{th}(t)$ este declanşată de *“first ack”*;
    - “Slow Start”: dacă $W(t)<W_{th}(t)$ fiecare “*first ack*” determină incrementarea ferestrei cu 1;
    - “Congestion Avoidance”: dacă $W(t)>W_{th}(t)$ fiecare “*first ack*” determină incrementarea ferestrei cu 1/W(t);
    - “Time out” la un anumit moment: face ca W(t) să fie setat la 1, $W_{th}(t)$ la W(t)/2, iar retransmisia porneşte de la A(t);

Protocolul TCP – operaţii de bază

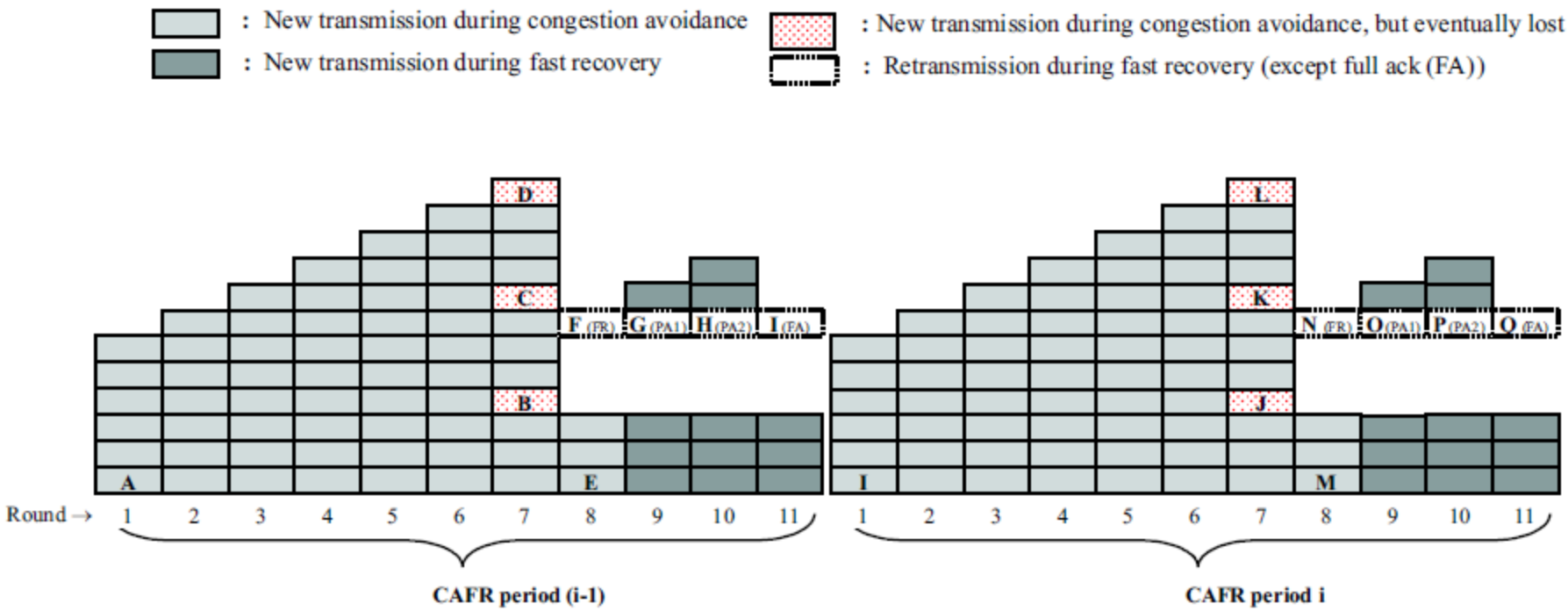


- Recuperarea pachetelor pierdute:
  - Dacă un pachet se pierde A(t) şi W(t) se incrementează până când se recepţionează "*first ack*" pentru ultimul pachet corect; pentru un caz particular fie M fereastra de pierderea, adică transmiţătorul trimite până la pachetul A+M-1 – acest ultim pachet are un RTO asociat, care va expira la un moment dat;
- Diferitele variante de TCP diferă după cum are loc recuperarea pachetelor pierdute:
  - TCP OldTahoe *("time-out recovery"):* transmiţătorul transmite până la pachetul A+M-1, după care aşteaptă pentru un "*time-out*";
  - TCP Tahoe *("fast retransmit"):* se defineşte un parametru K pozitiv, de valoare redusă, tipic K=3; dacă transmiţătorul primeşte K *ack* duplicate atunci nu se mai aşteaptă *time-out* ci se începe procedura de retransmisie conform algoritmului de bază;

Protocolul TCP – operaţii de bază

- TCP Reno *("fast retransmit; fast (but conservative)recovery")*: *fast-retransmit* implementat ca şi în cazul TCP Tahoe; dacă al K-lea *ack* duplicat se recepţionează la momentul t_0 atunci: $W(t_{0+})=M/2+K$, $W_{th}(t_{0+})=M/2$;
- TCP NewReno *("fast retransmit; fast recovery")*: după recepţia a K *ack* duplicate, este retransmis primul pachet eronat şi după recepţia primului *ack* parţial este retranmis următorul pachet pierdut; după ce se aşteaptă un număr de K *ack* duplicate packetele pierdute se recuperează în K etape succesive;
  - Se introduce noţiunea de *"full ack"* : generează *ack* pentru toate datele care sunt în tranzit la începutul etapei de "*fast retransmit*";
  - Se introduce noţiunea de "*partial ack*" : generează *ack* doar pentru o parte din pachetele care sunt în tranzit la începutul etapei de *"fast retransmit"*;

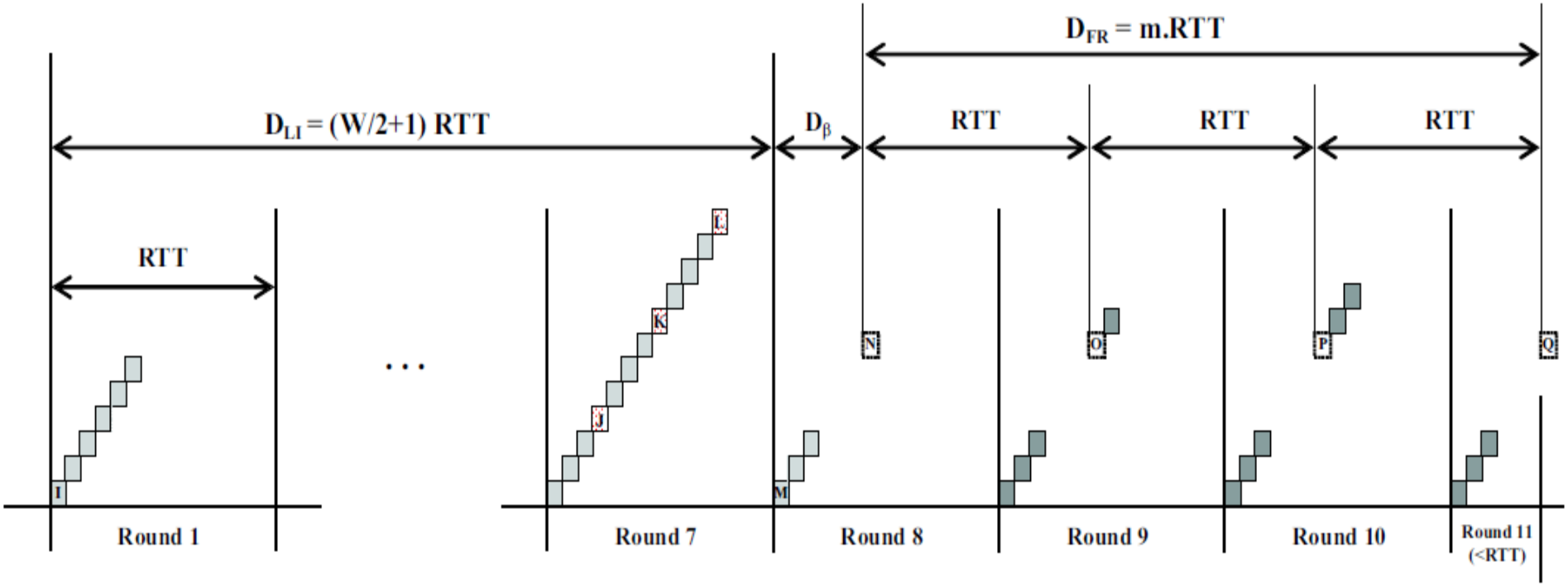
Protocolul TCP NewReno - detalii



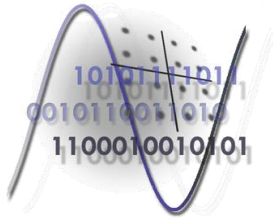
- CAFR – congestion avoidance
- FR – fast retransmit
- PA – partial acknowledge
- FA – full acknowledge

Protocolul TCP NewReno - detalii

- TCP NewReno - modul de lucru fast retransmit:
  - 1) se retransmite pachetul pierdut;
  - 2) se setează $W_{th}(t)=W(t)/2$;
  - 3) se setează $W(t)=W_{th}(t)+3$;

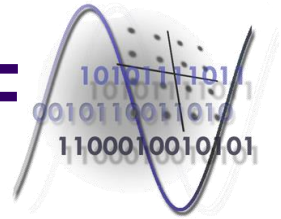


Protocolul FEC-TCP bazat pe coduri DF



- Dezavantajul major al TCP:
  - **Toate** pierderile de pachete sunt considerate congestie, ceea ce duce la o scădere a throughput-ului;
  - Când protocolul se utilizează pe canale cu rată mare de pierdere a pachetelor atunci performanţele scad semnificativ;
  - Performanţele pot fi îmbunătăţite substanţial prin utilizarea unei metode care poate distinge pierderile de pachete generate de erorile de pe linie de cele generate de congestie;
- FEC-TCP bazat pe coduri DF are următoarele caracteristici de bază:
  - Utilizează algoritmii de *“slow start”* şi *“congestion avoidance”* caracteristice protocoalelor TCP;
  - Pentru fiecare ACK se calculează RTT mediu şi dispersia;

Protocolul FEC-TCP bazat pe coduri DF



- După fiecare ACK recalculează un *"smooted RTT"*:

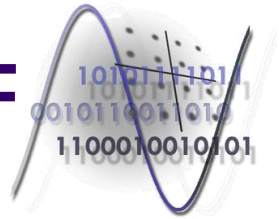
$$sRTT_{new} = \alpha \cdot sRTT_{old} + (1-\alpha)\overline{RTT}; \quad \alpha \in (0,1)$$

- Se calculează un interval *"time-out"* pe baza *"smooted RTT"*:

$$\Delta T_{TO} = (1+\beta)(sRTT + \sigma_{RTT}); \qquad \beta \in \mathbf{R}^{+}$$

- Pentru fiecare segment care este injectat în reţea se porneşte un timer având durata ΔT_{TO}; dacă timerul expiră înainte să sosească un *ACK* se efectuează un algoritm *"slow start"* sau *"congestion avoidance"* caracteristic protocolului TCP;
- *ACK*-urile pentru pachetele transmise sunt cumulative; *ACK* pentru pachetul *n* validează toate pachetele anterioare inclusiv pachetul *n*;
- Pachetele injectate în reţea sunt simboluri codate generate de codorul DF;

Protocolul FEC-TCP bazat pe coduri DF



- Alegerea corespunzătoare a parametrului ΔT_{TO} poate asigura separarea dintre pierderile de pachete generate de congestie respectiv de erorile de pe linie, pe baza următoarelor premize:
  - Erorile de pe canal generează de regulă pierderi de pachete independente;
  - Congestiile din reţea generează de regulă pierderi de pachete consecutive;
- Ideea de bază este de a “ascunde” pachetele pierdute de mecanismul de control a congestiei;
  - Protocolul nu reduce throughput-ul atunci când apar erori pe canale;
  - Congestia se poate detecta corect;

Protocolul FEC-TCP bazat pe coduri DF

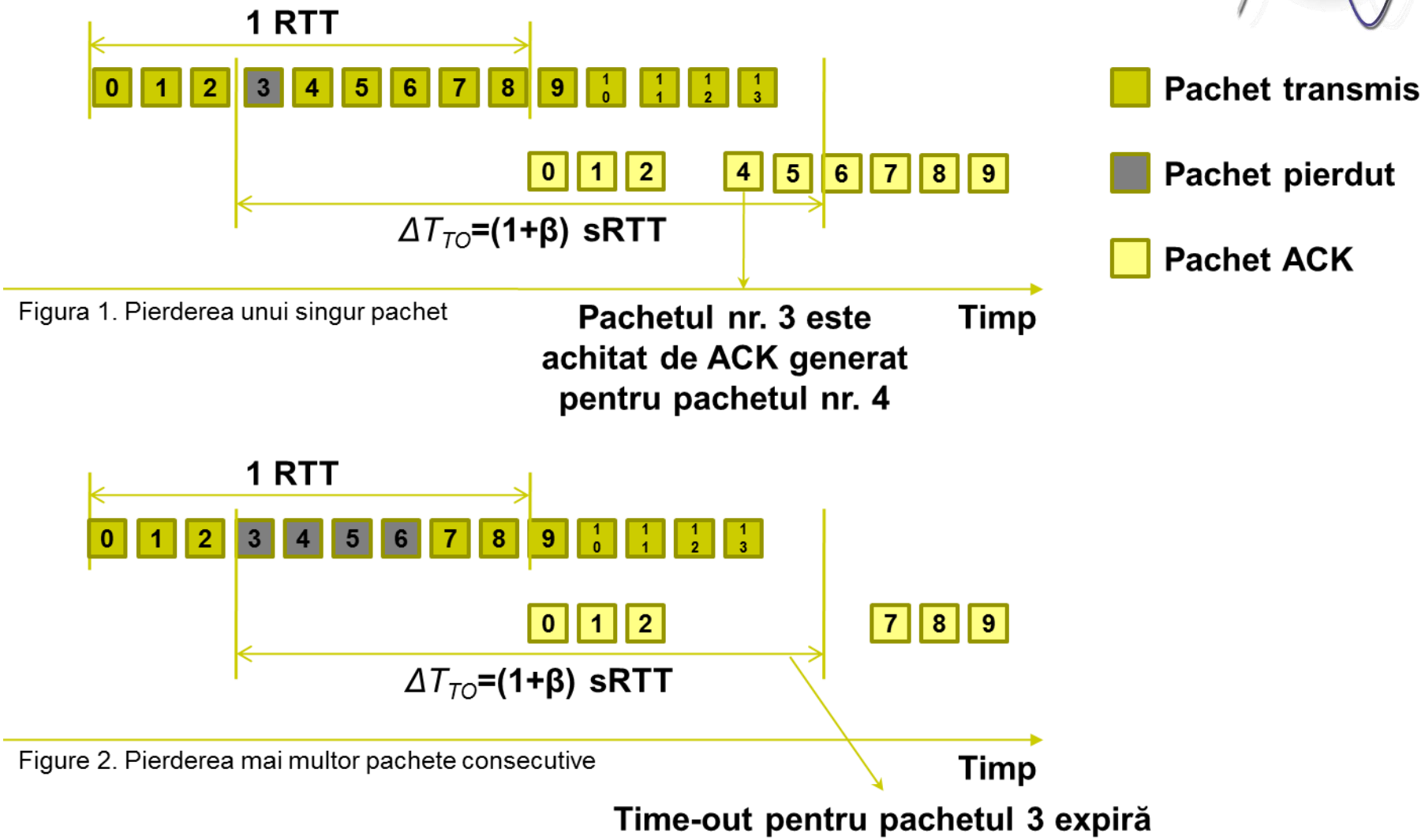


Figura 1. Pierderea unui singur pachet

Figure 2. Pierderea mai multor pachete consecutive

Modelul de canal utilizat

- **Modelul de canal Neyman**
- Modelează distribuţia erorilor pe un canal cu un număr redus de mecanisme care generează erori; este corespunzător pentru un model de erori generic;

$$P(r) = \frac{M_2^r}{r!} e^{-M_1} \sum_{j=0}^{\infty} \frac{z^j j^r}{j!}$$

unde: $z = M_1 e^{-M_2}$

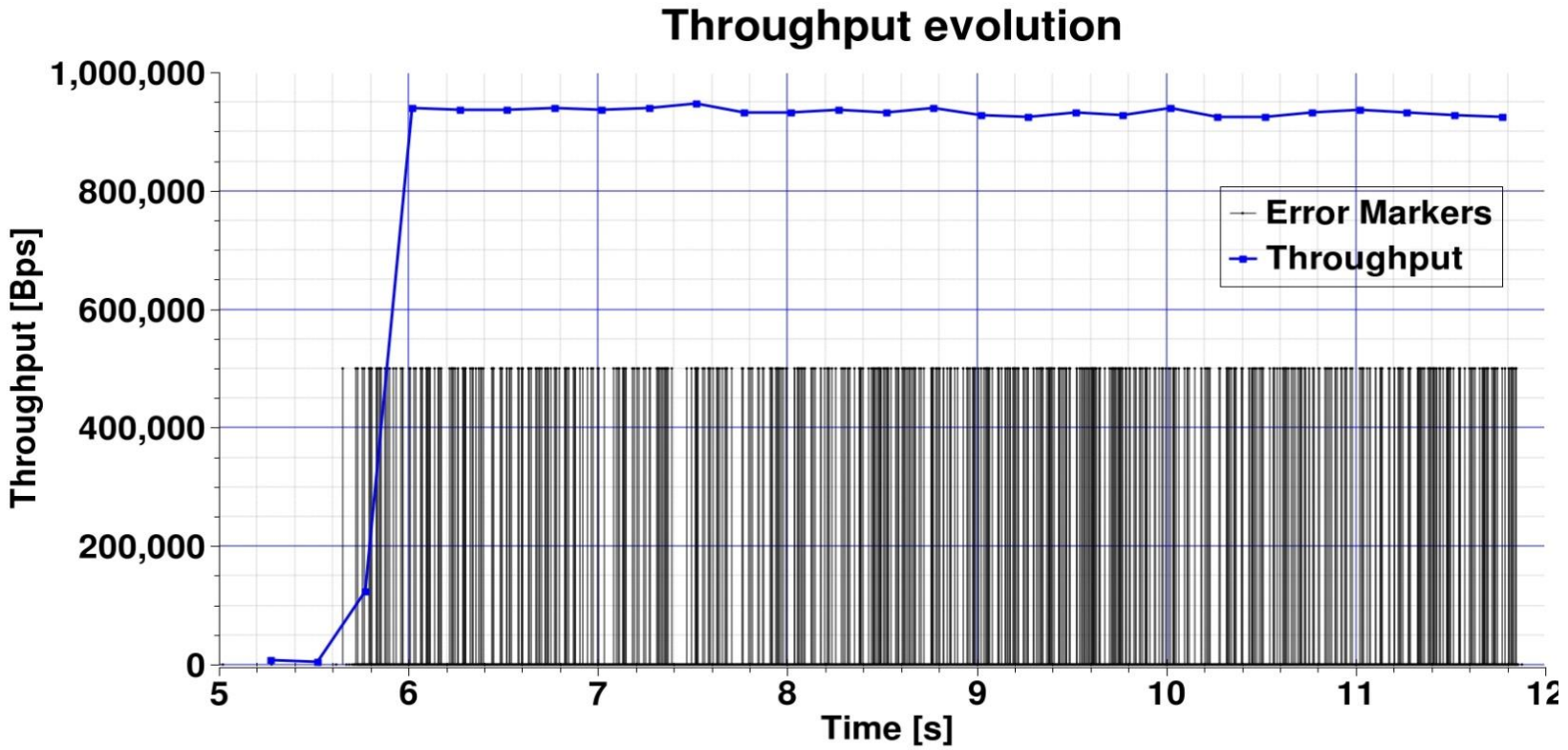
M_1 – numărul mediu de pachete de erori pe pachet/grup de biţi transmişi;

M_2 – număr mediu de erori pe pachet de erori;

  - Valoarea medie a distribuţiei este $M_1 \cdot M_2$, şi este egal cu $n \cdot p$, unde n este numărul total de biţi dintr-un pachet, iar p este media pe termen lung a probabilităţii de eroare pe bit;
- $P(r)$ este probabilitatea ca un pachet de n biţi să conţină exact r erori;

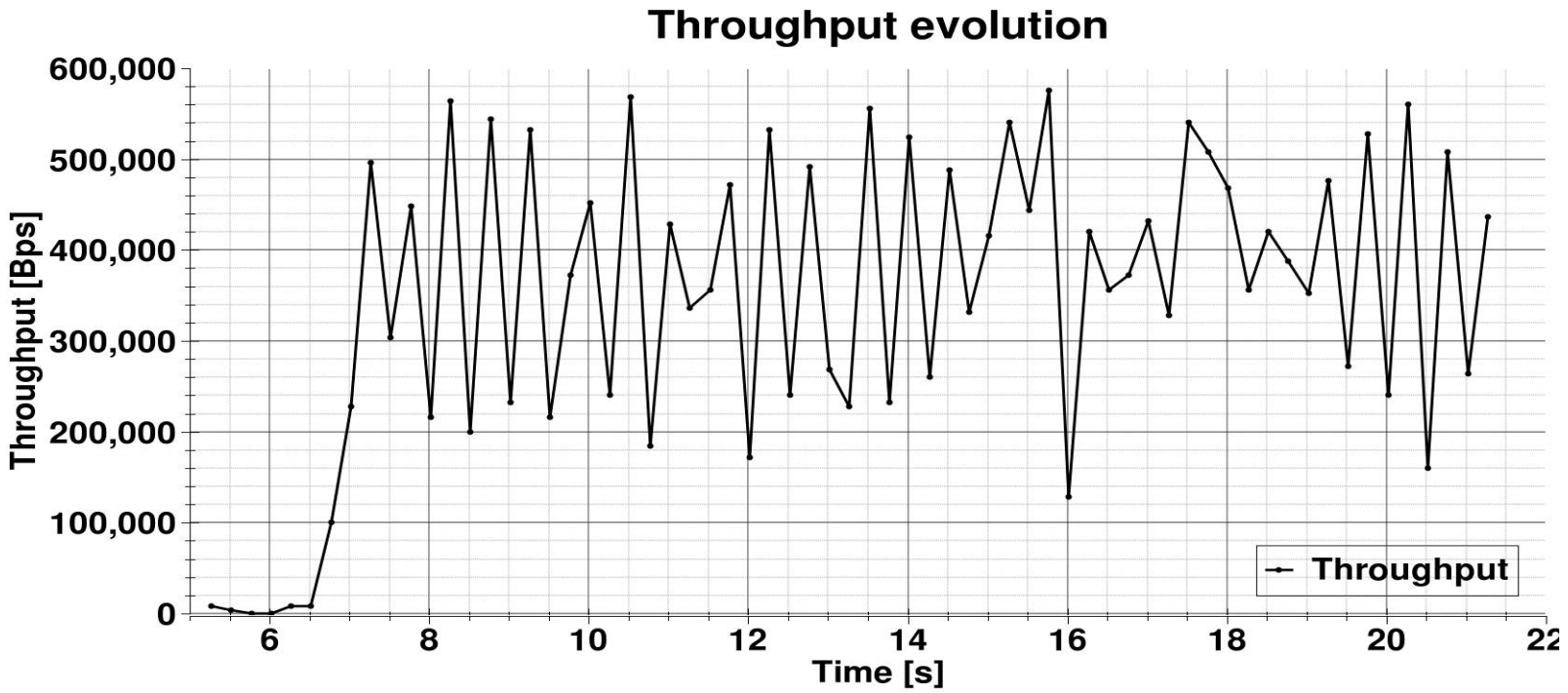
Evaluarea protocolului FEC-TCP

- Evaluările teoretice şi rezultatele simulărilor arată că protocolul este destul de insensibil la probabilităţi mari de pierdere a pachetelor;
- Evoluţia throughput-ului în timp nu este afectată (semnificativ) de erorile de pe canal;



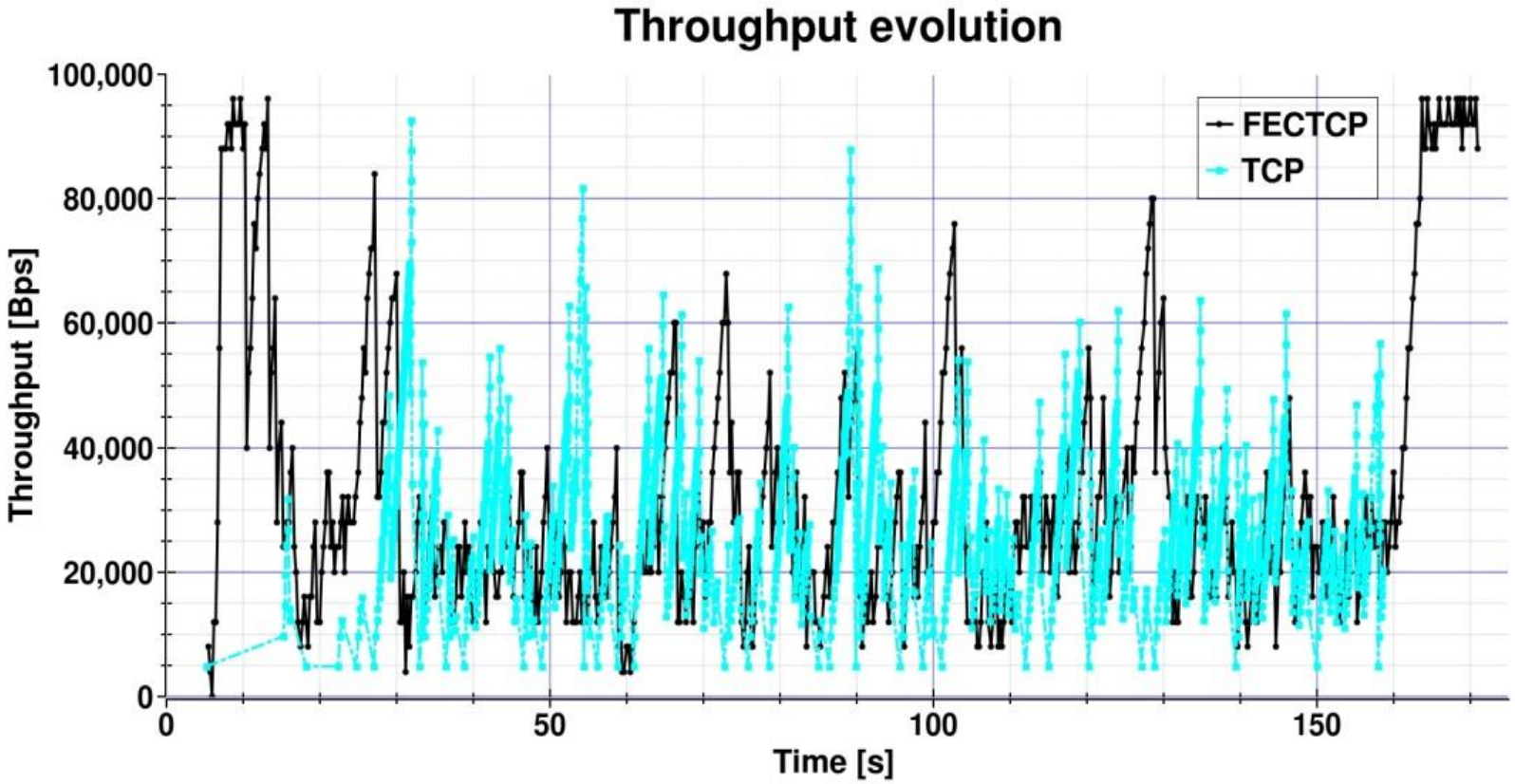
Evaluarea protocolului FEC-TCP

- Protocolul reduce throughput-ul în mod corespunzător când se detectează congestie în reţea;
- După reducerea throughput-ului acesta începe să crească din nou până când se detectează din nou congestie;



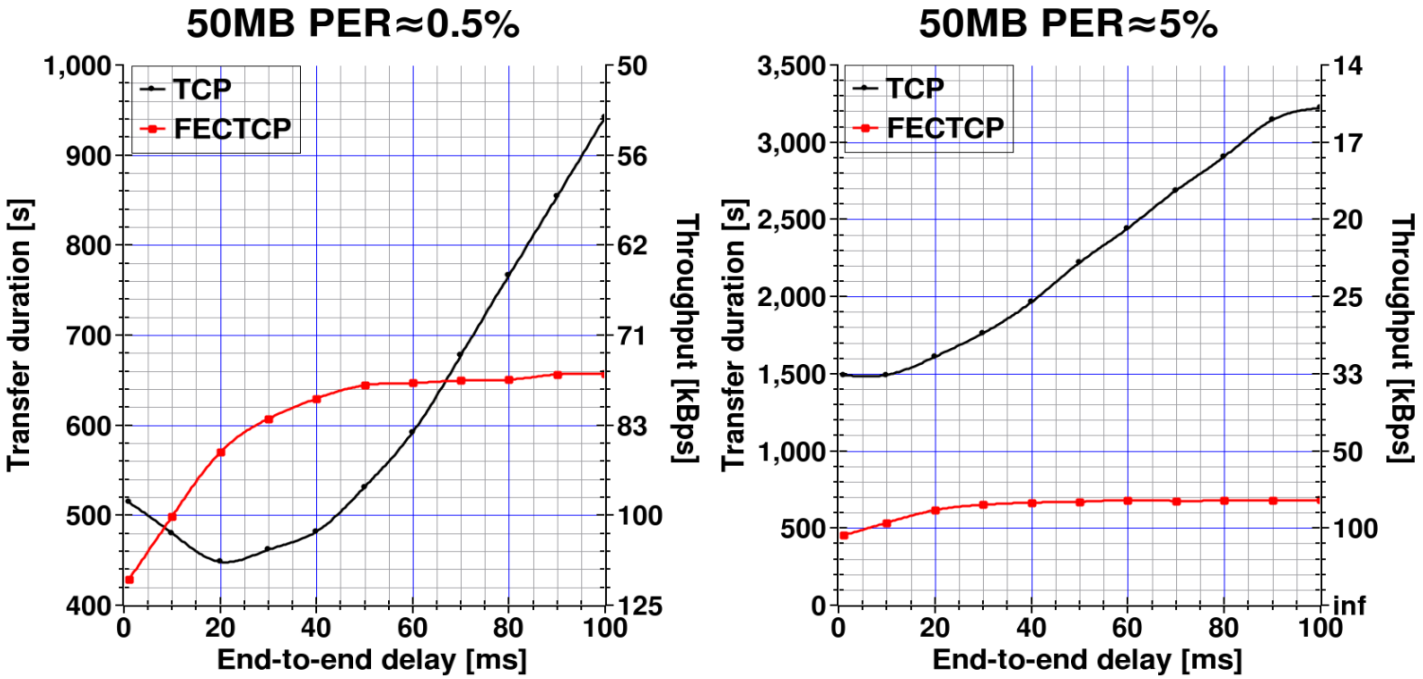
Evaluarea protocolului FEC-TCP

- Protocolul poate partaja resursele cu alte fluxuri din reţea;
  - Protocolul FEC-TCP este *“network friendly”*;



Evaluarea protocolului FEC-TCP

- Performanţele TCP sunt degradate sever de întârzierile mari “end-to-end” datorită dimensiunii finite a ferestrei de congestie;
  - Pentru întârzieri mici throughput-ul este limitat de PER;
  - Pentru întârzieri mari limitările sunt generate de dimensiunea finită a ferestrei de congestie;
- Degradarea de throughput nu este prezentă la FEC-TCP;



Evaluarea protocolului FEC-TCP

- Throughput-ul TCP se degradează sever când probabilitatea de pierdere a pachetelor depăşeşte un anumit prag;
- FEC-TCP este afectat numai moderat de către o probabilitate mare de pierdere a pachetelor;
  - Pentru probabilităţi de pierdere a pachetelor mai mari de 2% TCP este depăşit de FEC-TCP;