Modeling TCP Throughput: A Simple Model and its Empirical Validation

Ming-Wei Wu (gis89512@cis.nctu.edu.tw)
Department of Computer and Information Science
National Chiao Tung University
2000.10.24

Outline

- Introduction
- Model of TCP congestion control
- Derive a new analytic TCP throughput
  as a function of loss rate and average round trip time
- Compare with real-world TCP flows
- Discusses the assumptions underlying the model
  and related issue
- Conclusion

Introduction

The throughput of TCP's congestion control mechanism

Before:

As a function of packet loss and round trip delay.

- only the behavior of fast retransmit

Now:

As a function of loss rate and round trip delay

- plus the effect of TCP's timeout
- more accurate over a wider range of loss rates

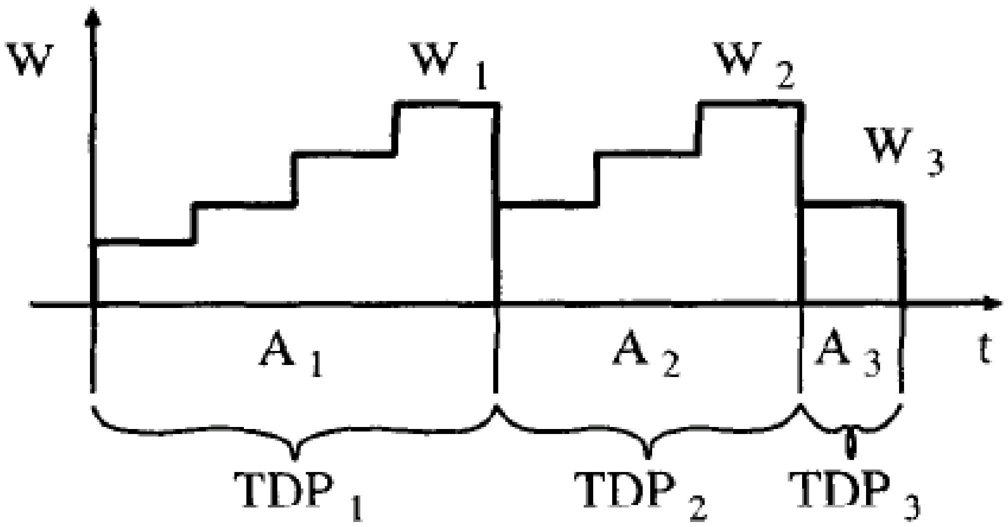
A Model for TCP Congestion Control Part I

Focus:

- Congestion avoidance behavior of TCP
  and its impact on throughput
- ACK behavior, packet loss indication is inferred by
  - duplicate ACK, triple-duplicate (TD)
  - timeout (TO)
- Reno flavor of TCP, by far most popular

A Model for TCP Congestion Control Part II

- Window size over time when loss indications are TD ACKs



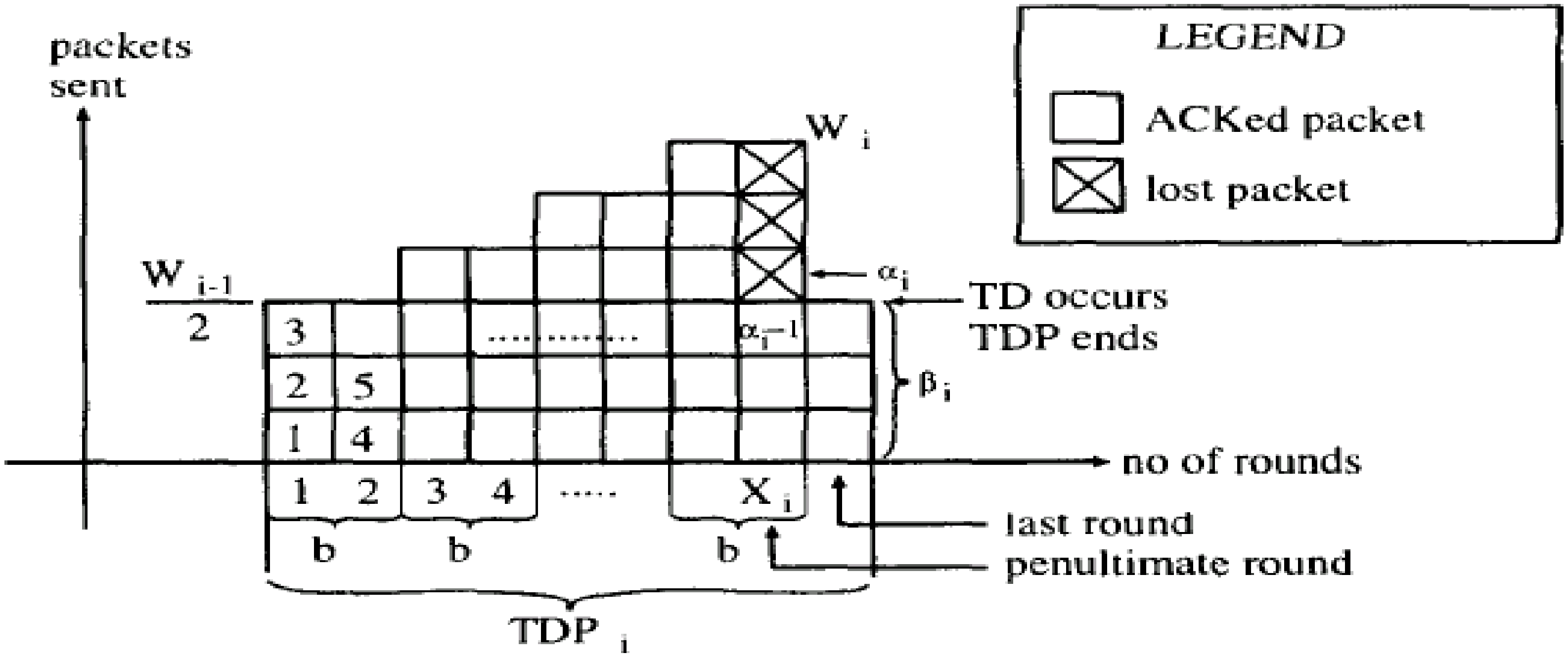
- No loss: +1/b packets per round
- With loss: /2

A_i 是 i-th TD Period 持續的時間長短

W_i 是 i-th TD Period 結束時的 Window 大小

Loss indications are exclusively "triple-duplicate" ACKs

- Packets sent during a TD period

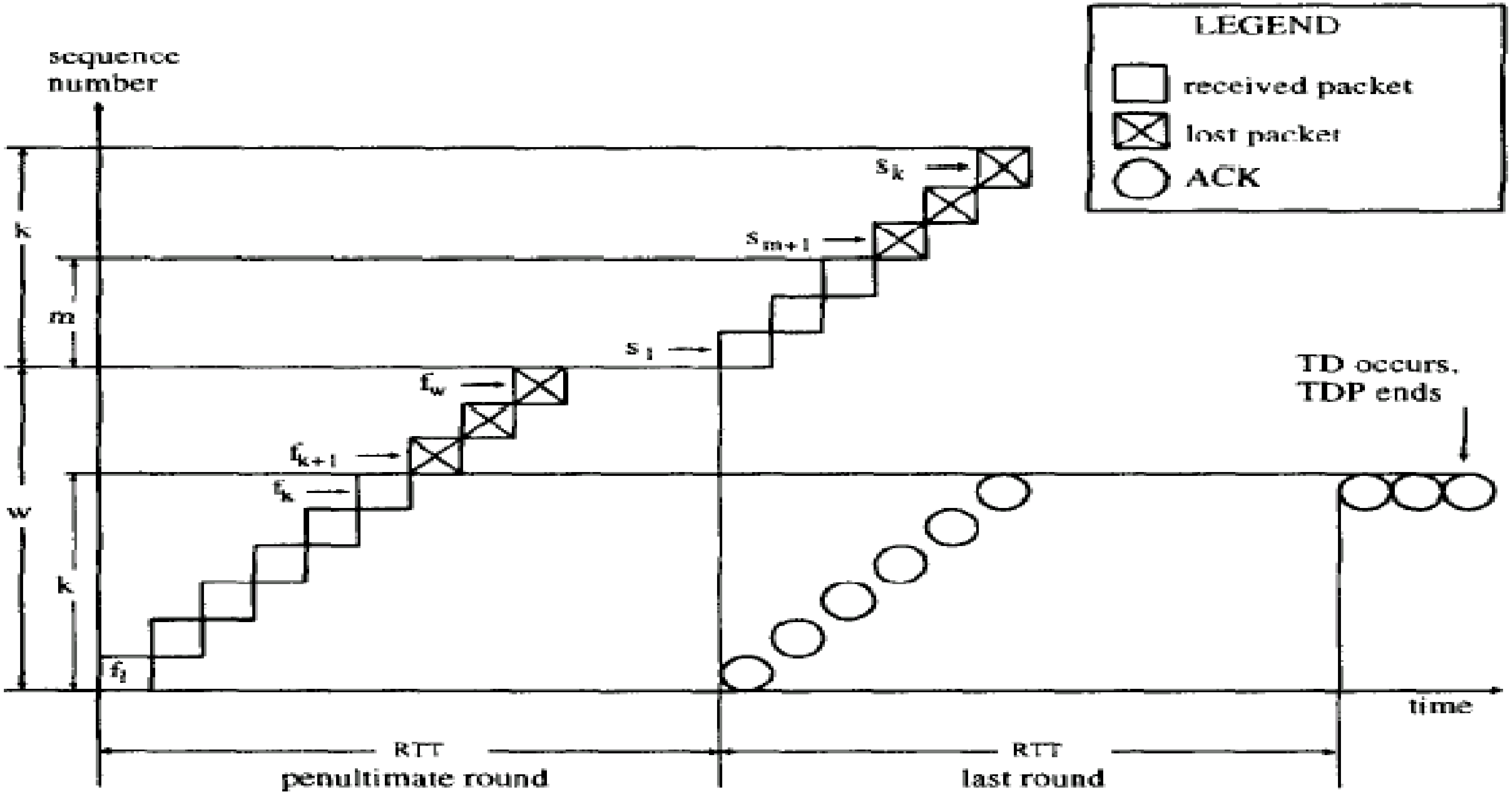


Y_i 是 i-th TD Period 之間所送的封包數

B 是 long-term steady-state 的 TCP throughput 可以表示成 $B = \frac{E[Y]}{E[A]}$

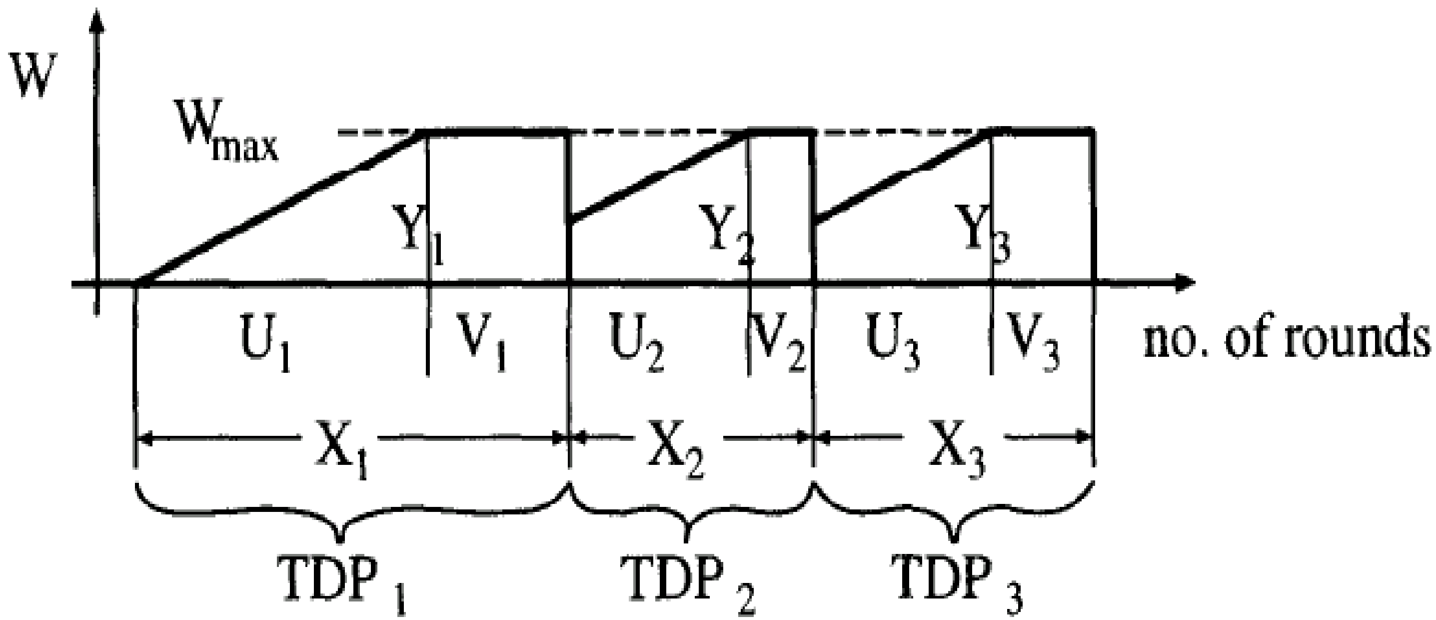
Loss indications are triple-duplicate ACKs and timeouts

- Packets and ACK transmissions preceding a loss indication



The impact of window limitation

- Fast retransmit with window limitation



Derive E[Y]

α_i 是 i-th TD Period 發生第一個 packet lost 爲止的封包數

X_i 是α_i 發生時的第幾個 Round 數

$$E[Y] = E[\alpha] + E[W] - 1$$

∵ 每個 round 的 packet lost 是 independent of 其他 round 的 packet lost

∴ $\{\alpha_i\}_i$ 是一串 independent 且 identical distributed (i.i.d.)的 random variables

$$P[\alpha = k] = (1-p)^{k-1} p, \quad k = 1, 2, \ldots$$

則α 的 mean 爲

$$E[\alpha] = \sum_{k=1}^{\infty} (1-p)^{k-1} pk = \frac{1}{p}$$

遂可整理原式成：

$$\boxed{E[Y] = \frac{1-p}{p} + E[W]}$$

Derive E[A]

定義 r_{ij} 是 i-th TD Period 的 j-th round 持續的時間長短

則 i-th TD Period 持續的時間長短，可以表示為 $A_i = \sum_{j=1}^{X_i+1} r_{ij}$

若考量 r_{ij} 本身是 random variables，且假設與 congestion window 的大小無關，

所以也 independent of round number，j。

那麼 $E[A] = (E[X]+1)E[r]$ 其中 $RTT = E[r]$

Derive E[W]

假設：$\frac{W_{i-1}}{2}$ 和 $\frac{X_i}{b}$ 都是整數

Window 大小的增加是介於 $\frac{W_{i-1}}{2}$ 和 W_i 之間，呈 $1/b$ 斜率的線性關係，

所以 $W_i = \frac{W_{i-1}}{2} + \frac{X_i}{b}, \quad i = 1,2,\ldots$

β_i 是 i-th TD Period 上一個 round 所送的封包數

$$Y_i = \sum_{k=0}^{X_i/b-1} \left(\frac{W_{i-1}}{2} + k \right) b + \beta_i$$

$$= \frac{X_i W_{i-1}}{2} + \frac{X_i}{2} \left(\frac{X_i}{b} - 1 \right) + \beta_i$$

$$= \frac{X_i}{2} \left(\frac{W_{i-1}}{2} + W_i - 1 \right) + \beta_i$$

→

$$\frac{1-p}{p} + E[W] = \frac{E[X]}{2} \left(\frac{E[W]}{2} + E[W] - 1 \right) + E[\beta]$$

⟹

$$E[W] = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b} \right)^2}$$

After all…

Only TD ACKs

$$B(p) = \frac{1}{RTT}\sqrt{\frac{3}{2bp}} + o(1/\sqrt{p})$$

TD ACKs and Timeouts

$$B(p) \approx \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p(1+32p^2)}$$

Plus window limitation

$$B(p) = \begin{cases} \dfrac{\frac{1-p}{p} + E[W] + \hat{Q}(E[W])\frac{1}{1-p}}{RTT(\frac{b}{2}E[W_u]+1) + \hat{Q}(E[W])T_0\frac{f(p)}{1-p}} & \text{if } E[W_u] < W_{max} \\ \dfrac{\frac{1-p}{p} + W_{max} + \hat{Q}(W_{max})\frac{1}{1-p}}{RTT(\frac{b}{8}W_{max} + \frac{1-p}{pW_{max}} + 2) + \hat{Q}(W_{max})T_0\frac{f(p)}{1-p}} & \text{otherwise} \end{cases}$$

Measurement and Trace Analysis

Sender	Receiver	Packets Sent	Loss Indic.	TD	TO	RTT	Time Out
manic	alps	54402	722	19	703	0.207	2.505
manic	baskerville	58120	735	306	429	0.243	2.495
manic	ganef	58924	743	272	471	0.226	2.405
manic	mafalda	56283	494	2	492	0.233	2.146
manic	maria	68752	649	1	648	0.180	2.416
manic	spiff	117992	784	47	737	0.211	2.274
manic	sutton	81123	1638	988	650	0.204	2.459
manic	tove	7938	264	1	263	0.275	3.597
void	alps	37137	838	7	831	0.162	0.489
void	baskerville	32042	853	339	514	0.482	1.094
void	ganef	60770	1112	414	696	0.254	0.637
void	maria	93005	1651	33	1618	0.152	0.417
void	spiff	65536	671	72	599	0.415	0.749
void	sutton	78246	1928	840	1088	0.211	0.601
void	tove	8265	856	5	843	0.272	1.356
babel	alps	13460	1466	0	1461	0.194	1.359
babel	baskerville	62237	1753	197	1556	0.253	0.429
babel	ganef	86675	2125	398	1727	0.201	0.306
babel	spiff	57687	1120	0	1120	0.331	0.953
babel	sutton	83486	2320	685	1635	0.210	0.705
babel	tove	83944	1516	1	1514	0.194	0.520
pif	alps	83971	762	0	760	0.168	7.278
pif	imagine	44891	1346	15	1329	0.229	0.700
pif	manic	34251	1422	43	1377	0.257	1.454

Table 2: Summary data from 1hr traces

Discussion of the Model and the Experimental Results

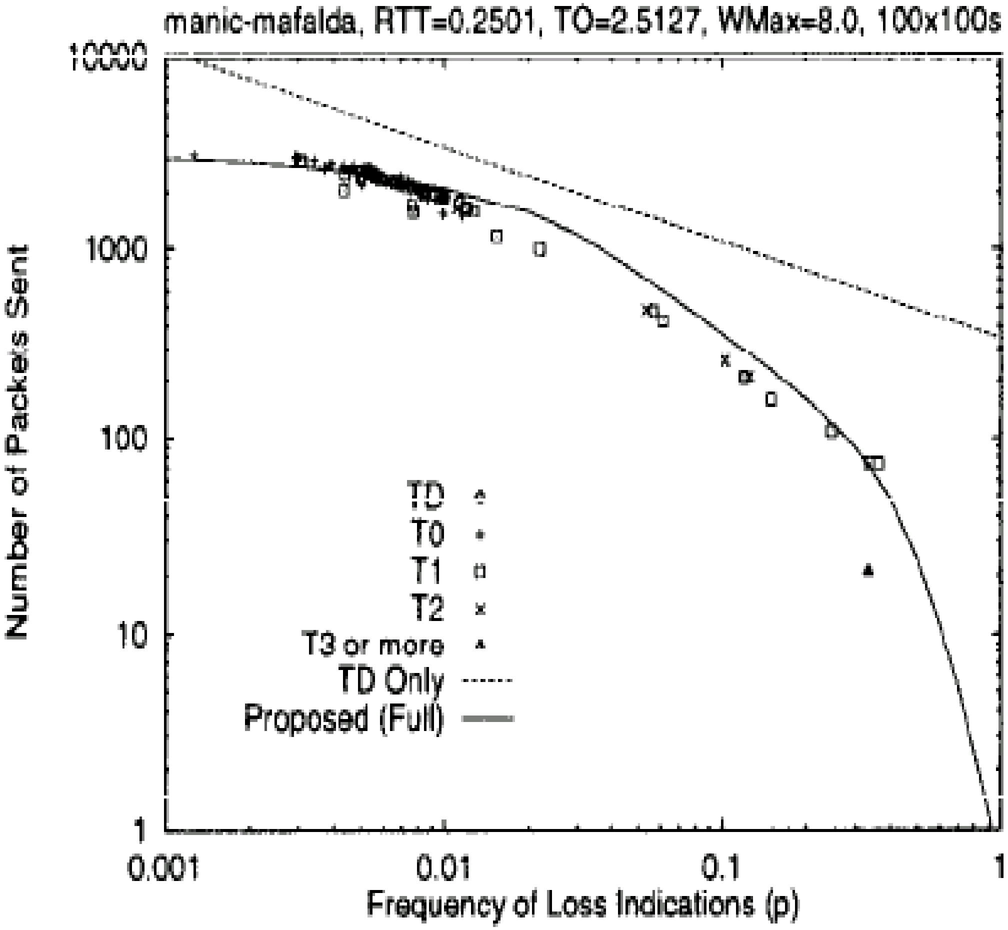


Figure 14: manic to mafalda

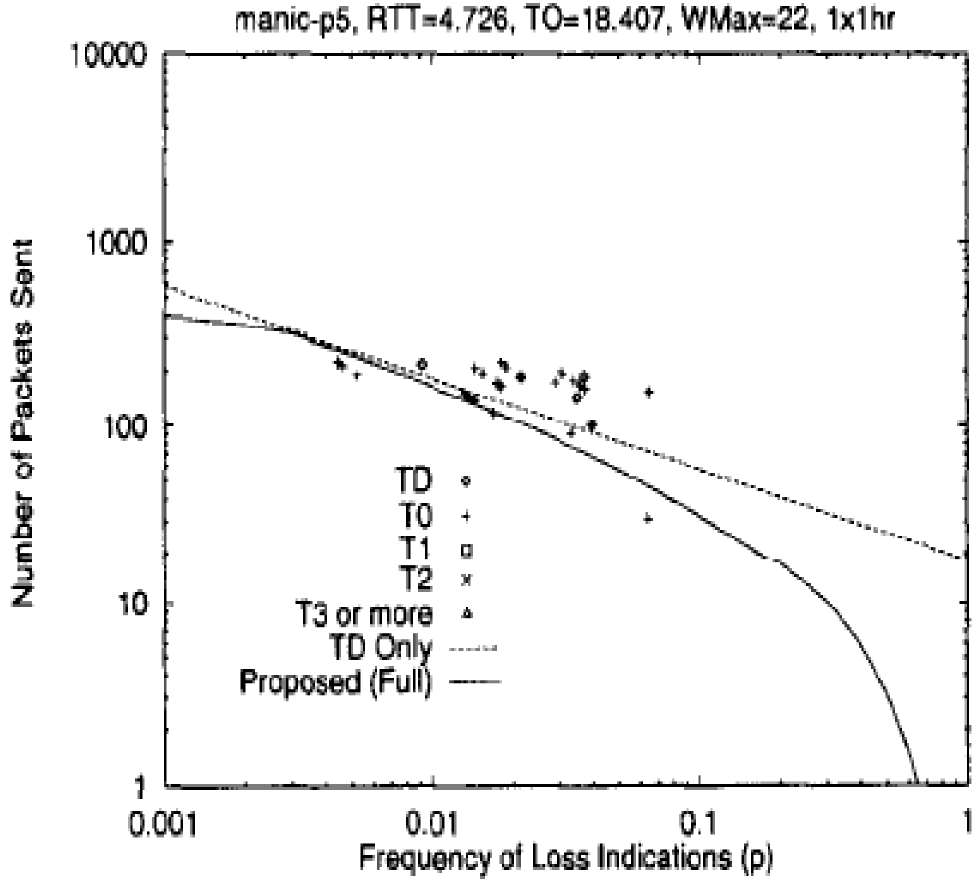


Figure 21: manic to p5

Discussion of the Model and the Experimental Results: Part I

Observations:

- timeout → a significant fraction of total loss indication
- Frequent exponential back off due to multiple timeouts
- (TD only case)在Frequency of loss indication (p)
  超過 5%就不太準

Neglect:

- fast recovery
- slow start

Discussion of the Model and the Experimental Results: Part II

Assumptions:

- packet losses within a round are correlated (drop-tail policy); independent of losses in other rounds
- RTT is independent of the window size (but modem line has some problem, coefficient is as high as 0.97)
- all senders implement TCP-Reno ; consider OS differences as well.
  (ex. Linux uses two duplicate ACKs, exponential backoff does not exactly follow the standard algorithm)

Conclusion

- former mechanisms significantly overestimate throughput.
- Timeout has a significant impact on the TCP performance.
- Future works:
  - account for the effect of fast recovery and fast retransmit
  - investigate the behavior of TCP over slow links with dedicated buffer
  - more relaxed packet lost policy, modified to incorporate a loss distribution function