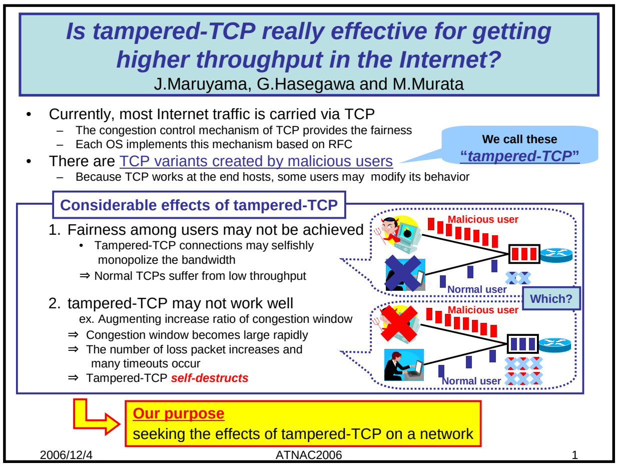


Is tampered-TCP really effective for getting higher throughput in the Internet?

J.Maruyama, G.Hasegawa and M.Murata

- Currently, most Internet traffic is carried via TCP
  - The congestion control mechanism of TCP provides the fairness
  - Each OS implements this mechanism based on RFC
- There are TCP variants created by malicious users
  - Because TCP works at the end hosts, some users may modify its behavior

We call these **“*tampered-TCP*”**

Considerable effects of tampered-TCP

1. Fairness among users may not be achieved
   - Tampered-TCP connections may selfishly monopolize the bandwidth

   ⇒ Normal TCPs suffer from low throughput
2. tampered-TCP may not work well

   ex. Augmenting increase ratio of congestion window

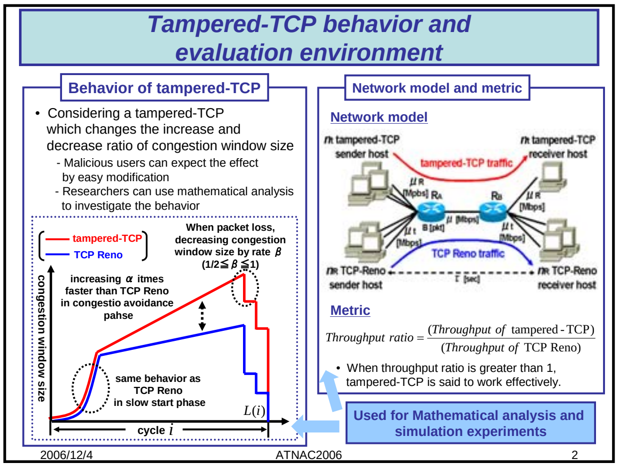
   ⇒ Congestion window becomes large rapidly

   ⇒ The number of loss packet increases and many timeouts occur

   ⇒ Tampered-TCP ***self-destructs***

Our purpose

seeking the effects of tampered-TCP on a network



Tampered-TCP behavior and evaluation environment

Behavior of tampered-TCP

- Considering a tampered-TCP which changes the increase and decrease ratio of congestion window size
  - Malicious users can expect the effect by easy modification
  - Researchers can use mathematical analysis to investigate the behavior

tampered-TCP
TCP Reno
When packet loss, decreasing congestion window size by rate β (1/2≦β≦1)
increasing α itmes faster than TCP Reno in congestio avoidance pahse
congestion window size
same behavior as TCP Reno in slow start phase
L(i)
cycle i

Network model and metric

Network model

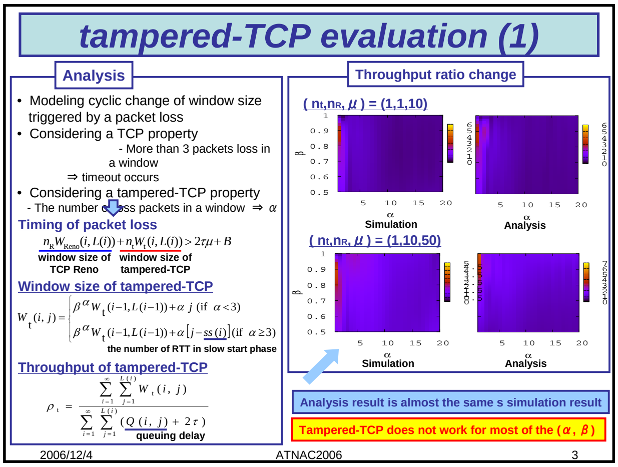
nt tampered-TCP sender host
nt tampered-TCP receiver host
tampered-TCP traffic
μR [Mpbs]
RA
RB
μR [Mbps]
μ [Mbps]
B [pkt]
μt [Mbps]
μt [Mbps]
TCP Reno traffic
nR TCP-Reno sender host
nR TCP-Reno receiver host
τ [sec]

Metric

$$\textit{Throughput ratio} = \frac{(\textit{Throughput of}\ \text{tampered-TCP})}{(\textit{Throughput of}\ \text{TCP Reno})}$$

- When throughput ratio is greater than 1, tampered-TCP is said to work effectively.

Used for Mathematical analysis and simulation experiments



tampered-TCP evaluation (1)

Analysis

- Modeling cyclic change of window size triggered by a packet loss
- Considering a TCP property
  - More than 3 packets loss in a window

  ⇒ timeout occurs
- Considering a tampered-TCP property
  - The number of loss packets in a window ⇒ α

Timing of packet loss

$$\underbrace{n_R W_{\text{Reno}}(i, L(i))}_{\text{window size of TCP Reno}} + \underbrace{n_t W_t(i, L(i))}_{\text{window size of tampered-TCP}} > 2\tau\mu + B$$

Window size of tampered-TCP

$$W_t(i,j) = \begin{cases} \beta^{\alpha} W_t(i-1, L(i-1)) + \alpha\, j\ (\text{if}\ \ \alpha < 3) \\ \beta^{\alpha} W_t(i-1, L(i-1)) + \alpha\left[j - \underbrace{ss(i)}_{\text{the number of RTT in slow start phase}}\right] (\text{if}\ \ \alpha \geq 3) \end{cases}$$

Throughput of tampered-TCP

$$\rho_t = \frac{\sum_{i=1}^{\infty} \sum_{j=1}^{L(i)} W_t(i, j)}{\sum_{i=1}^{\infty} \sum_{j=1}^{L(i)} (\underbrace{Q(i, j)}_{\text{queuing delay}} + 2\tau)}$$

Throughput ratio change

(nt,nR, μ) = (1,1,10)

Simulation

Analysis

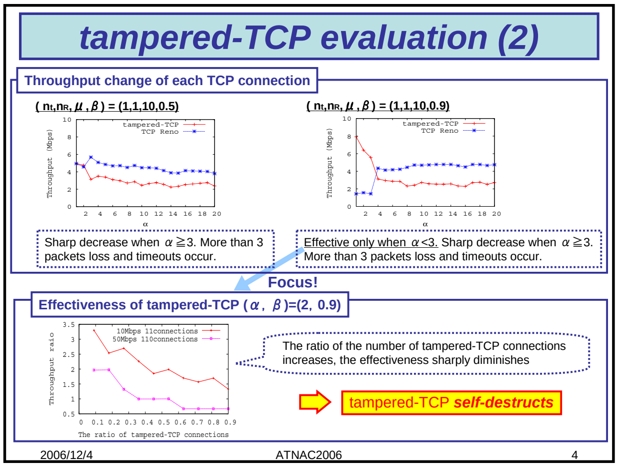
(nt,nR, μ) = (1,10,50)

Simulation

Analysis

Analysis result is almost the same s simulation result

Tampered-TCP does not work for most of the (α, β)



tampered-TCP evaluation (2)

Throughput change of each TCP connection

(nt,nR, μ, β) = (1,1,10,0.5)

Throughput (Mbps)
0 2 4 6 8 10
tampered-TCP
TCP Reno
2 4 6 8 10 12 14 16 18 20
α

Sharp decrease when α≧3. More than 3 packets loss and timeouts occur.

(nt,nR, μ, β) = (1,1,10,0.9)

Throughput (Mbps)
0 2 4 6 8 10
tampered-TCP
TCP Reno
2 4 6 8 10 12 14 16 18 20
α

Effective only when α<3. Sharp decrease when α≧3. More than 3 packets loss and timeouts occur.

Focus!

Effectiveness of tampered-TCP (α, β)=(2, 0.9)

Throughput raio
0.5 1 1.5 2 2.5 3 3.5
10Mbps 11connections
50Mbps 110connections
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
The ratio of tampered-TCP connections

The ratio of the number of tampered-TCP connections increases, the effectiveness sharply diminishes

tampered-TCP ***self-destructs***