

# ROBUST TCP: AN IMPROVEMENT ON TCP PROTOCOL

**SEIFEDDINE KADRY[1], ISSA KAMAR[1], ALI KALAKECH[2], MOHAMAD SMAILI[1]**
[1]Lebanese University - Faculty of Science, Lebanon
[1]Lebanese University - Faculty of Business, Lebanon
E-mail: skadry@gmail.com

**ABSTRACT**

The Transmission Control Protocol (TCP) is the most popular transport layer protocol for the internet. Congestion Control is used to increase the congestion window size if there is additional bandwidth on the network, and decrease the congestion window size when there is congestion.
This paper uses a classic TCP which we called Robust TCP with an accurate algorithm of congestion detection in order to improve the performance of TCP. Our TCP Robust only reacts when it receives an ECN (Explicit Congestion Notification) mark. The evaluation result shows a good performance in the terms of drop ratio and throughput.

**Keywords:** *Congestion Control, TCP, ECN, Implicit Congestion Notification.*

## I. INTRODUCTION

TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications. Congestion events in communication networks cause packet losses, and it's well known that these losses occur in burst.TCP congestion control involves two tasks:

1. Detect congestion
2. Limit Transmission rate

To achieve good performance and obtain a Robust TCP, it is necessary and important to control network congestion, by limiting the sending rate and regulating the size of congestion window (Cwnd) after the detection of congestion.TCP congestion control operates in a closed loop that infers network conditions and reacts accordingly by means of losses. A negative return is due to a loss of a segment which can be translated by decreasing the flow from the source through a reduction in the size of window control. TCP considers loss of a segment as a congestion in the network, the detection of this loss can be done in several ways: Timeout, Three Duplicate ACKs (Fast retransmit) and by receiving a partial ACK.

The state is:

- If Packet Loss or congestion event =>TCP decreases Cwnd.
- All is well and no congestion in the network, i.e., TCP increases Cwnd.

At all cases, loss indication should be done with accuracy because it may lead to false indications like: *Spurious retransmission.*
*Spurious timeout* occurs when a non lost packet is retransmitted due to a sudden RTT (Round Trip Time) increase (hand over, high delay, variability, rerouting . .) which implies to an expiration of the retransmission timer set with a previous and thus outdated RTT value.
This effect is known to be the root cause of spurious retransmission.
The function of the congestion control is an essential element to the stability of the internet.
Indeed, TCP congestion control reduces the flow when it detects a loss in the network. Therefore, it is important to be accurate in the loss detection to improve the performance of TCP.
A congestion event (or loss event) corresponds to one or several losses (or in the context of ECN: at least one acknowledgment path with an ECN-echo) occurring in one TCP window during one current RTT period, it means that a congestion event begins when the first loss occurs and finishes one RTT later.
In this paper, we propose a congestion detection algorithm that is realized independently of the

TCP code. To improve the TCP by reducing the Cwnd, we aim to illustrate the feasibility of the concept by demonstrating that we can both obtain similar performances and also improve the accuracy of the detection outside the TCP stack.
We implement the Implicit Congestion Notification (ICN) algorithm to better understand and investigate the problem of congestion events estimation.
This paper is organized as follows: section 2 presents related works, section 3 shows the architecture of the congestion detection, section 4 presents the detailed discussion for the Robust TCP with ICN congestion detection algorithm, and section 5 presents an evaluation of the TCP Robust using simulations.
Finally, section 6 concludes this article and presents some perspectives.

## II. RELATED WORKS:

Over the past few years, several solutions have been proposed to improve the performance of TCP. In [5] proposed TCP-DCR modifications to TCP's congestion control mechanism to make it more robust to non-congestion events, this is implemented by using the delay "tau" based on a timer.
Our mechanism is different; it relies on the accurate congestion detection algorithm (ICN) and uses the timestamp option to detect spurious timeout which can more improve the reliability of the algorithm and leads to a real Robust TCP.
In **Forward RTO-Recovery (F-RTO)**: the F-RTO algorithm of the TCP sender monitors the incoming acknowledgments to determine whether the timeout was spurious.
TCP suffers from the inaccuracy of the congestion detection in the other TCP agents, for this reason we design an accurate mechanism of congestion detection (ICN) that interacts with TCP robust.
Our study must prove the functionality of our TCP with ICN is better than other versions of TCP. For this point we have to show that the mechanism of congestion detection for some TCP variants (New-Reno, Sack) doesn't detect well when there is congestion and doesn't not work well more than TCP Robust with ICN.
In [5], the idea or the solution proposed for the detection of congestion is the delay of the time to infer congestion by T, and this value should be large to recover from non-congestion event, and should be small to avoid expensive RTO.
Our approach is different by using a classic TCP that responds only to an accurate algorithm of congestion detection.

## III. STAND-ALONE TCP CONGESTION EVENTS ALGORITHMS

In this section we present the architecture of decorrelating congestion Detection from the Transport Layer (figure 1).The main goal of this architecture is to simplify the task of kernel developers as well as improve TCP performances. This scheme opens the door to another way to react to congestion by enabling ECN emulation at end-host. In this case ICN emulates ECN marking to imply a congestion window reduction.

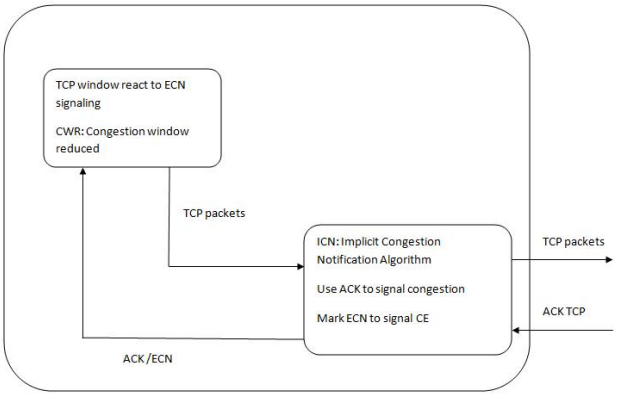


Figure. 1. Decorrelating Congestion Detecting from the Transport Layer

## IV. ROBUST TCP ALGORITHM:

Our proposed algorithm which we called Robust TCP is to make the congestion detection reliable and to distinguish the causes of losses in order to improve the flow control.
The main idea is to determine CE (i.e. the congestion detection) which impact on the TCP flow performance by monitoring the TCP flow itself.
The principle is to obtain a detection system at the edge of a network or at the sender side which analyses the TCP behavior through the observation of both data packets and acknowledgments paths.
So, the scenario is to make a new version of TCP (Robust TCP) without detection of congestion. Robust TCP doesn't reacts (reducing of Cwnd) whenever it doesn't receive a notification ECN.

Robust TCP must interact with ICN algorithm through ECN. Once we have congestion indication and the congestion event is validated, in this case it must notify the TCP we are exploring the functionality of Robust TCP and the ICN algorithm with the interaction between each other.

Robust TCP maintains all the functions of TCP Reno (slow start, Congestion avoidance, Fast retransmit and Fast recovery) and modified by adding error control and limited transmit (like in New-Reno TCP) to avoid unnecessary timeouts.

Robust TCP is a classic version of TCP but very sensitive to packet loss. It contains the major congestion control phases:

1. Slow start and congestion avoidance (increase Window size).
2. Fast retransmit (Detection of congestion).
3. Fast recovery.

1. Slow start:
   - When ACK received: cwnd++ which means for every ACK received, the sender sends two more segments.
   - Exponential increase in the window (Every RTT: cwnd = 2*cwnd)
   - Threshold (sstrhesh) controls the change to congestion avoidance.
2. Congestion avoidance
   - When ACK received: cwnd+ = 1/cwnd.
   - Linear increment of cwnd (every RTT: cwnd++) slow start is exists until cwnd is smaller or equal to ssthresh. Later congestion avoidance takes over.
3. Fast retransmit:

TCP generates duplicate ACK when out-of-order segments are received. In this case Fast retransmit uses "duplicate ACK" to trigger retransmission packets, so the sender does not wait until timeout for retransmission, sender retransmits the missing packet after receiving 3
DUPACK.

4. Fast recovery:

TCP retransmits the missing packet that was signaled by three duplicate ACKs and waits for an acknowledgment of the entire transmit window before returning to congestion avoidance. If there is no acknowledgment, TCP Robust experiences a timeout and enters the
Slow-start state.

TCP recovers much faster from fast retransmit than from timeout. When congestion window is small, the sender may not receive enough dupacks to trigger fast retransmit and has to wait for timer to expire but under
Limited transmit, sender will transmit a new segment after receiving 1 or 2 DUPACKs if allowed by receivers advertised window to generate more dupacks.

Robust TCP is poor in performance without detection of congestion and worse than other TCP like TCP New-Reno and Sack. It reacts only on the receiving of ECN notification.

Once it doesn't receive a notification that means there is no congestion control on TCP and the window keep increasing, but in case of receiving ECN that will indicate the occurrence of congestion indication notified by ICN, than Robust TCP reacts by limiting its sending rate and takes the full meaning of its name.

### IV.1 **ICN with Timestamp**

**ICN (implicit congestion notification)** is an algorithm for congestion detection implemented outside the TCP stack to analyze TCP flow and to better understand the problem of congestion events and than to conclude if the congestion occurs in the network or no and it is also more accurate in congestion detection than TCP.

The main goal of ICN is to determine the losses (i.e. the congestion detection) which impact on the TCP flow performance by observing the flow itself which mean by looking at the losses occurring over an RTT period given.

ICN is a generic algorithm that doesn't depend on the TCP version used which implements a congestion control where a negative feedback means a loss. It is important to note that ICN doesn't manage the error control which remains under the responsibility of TCP

Starting from the observation of the data segments and the acknowledgments, we identify each TCP connection with a state machine. This state machine indentifies the control congestion phase and classifies retransmission as spurious or not.TCP congestion control reacts following binary notification feedbacks allowing assessing whether the network is congested or not.

ICN algorithm consists of two states:

1. **Normal state**: which characterizes TCP connection without losses, in this state no congestion occurs and the sender receive the ACK normally.
2. **Congestion state:** This state starts from the loss of the first window data segment. When a loss occurs ICN enters in this state and waits to

the congestion event to be validated to notify Robust TCP about this loss. When the top of the window is acknowledged, ICN enters in the normal state.

To improve the performance of the congestion detection algorithm and especially against spurious timeout we added the timestamp option, in order once the congestion happens ICN enter in this state and append a timestamp to let the sender to compute the RTT estimate based on returned timestamp in ACK.

Time stamps used in this state to measure the round trip time (RTT) of a given TCP segment and including retransmitted segment, this option also can help to eliminate the retransmission ambiguity ( due to false indication) and identifies when retransmission is spurious or not.

Spurious Timeout are inevitable and not rare in data networks, for this reason and once the congestion event occurs, ICN enter in the congestion event state, timestamp is added for each data segment. Timestamp can be considered as an acknowledging mechanism in the time domain.

In the figure (2) shown below we will present the flowchart of TCP Robust with ICN mechanism:

### IV.2 Robust TCP and ICN interaction

ICN is an accurate congestion detection algorithm where after detecting a loss event in the congestion state, the congestion event (CE) must be validated.

The validation of CE should lead to a congestion indication which is the principle responsible to inform the Robust TCP about the congestion. The confirmation method due to a congestion indication is ECN (Explicit congestion notification), which is the main fag in the ACK to notify the loss to the source TCP. Once the source is signaled by ECN notification it reacts by reducing its window (Cwnd) and this time Robust TCP takes the full meaning of its name.

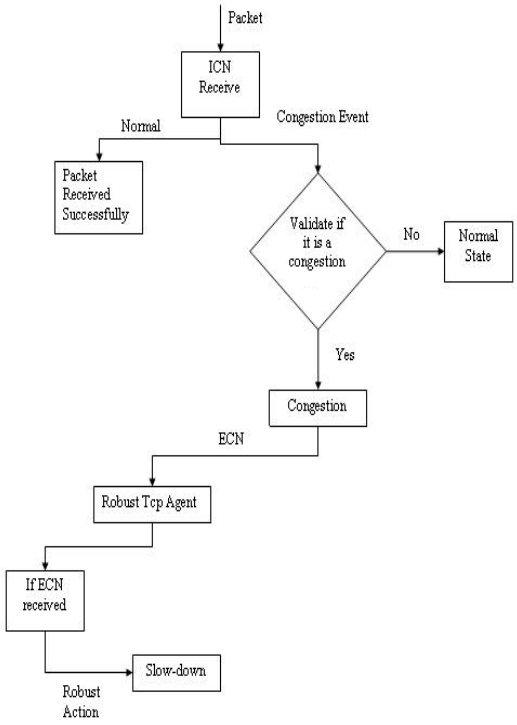


Figure 2: Robust TCP with ICN detection algorithm

After reducing its window, we can notice very well the decreasing of the number of dropped packets (d) in the network due to using of ICN congestion detector and our TCP becomes better in performance than others like
TCP New-Reno and Sack.

## V. VALIDATIONS AND EVALUATIONS

In this section we evaluate the performance of Robust TCP with ICN algorithm. The main idea is to build an algorithm of congestion detection outside the TCP stack that is responsible to detect the loss and notify it to Robust TCP.

The architecture of our tools is shown in the figure (3), which is mainly composed from the following components:

1. Network topology.
2. Traffic model.
3. Performance evaluation metrics.

After the simulation is done, a set of result statistics and graphs are generated.

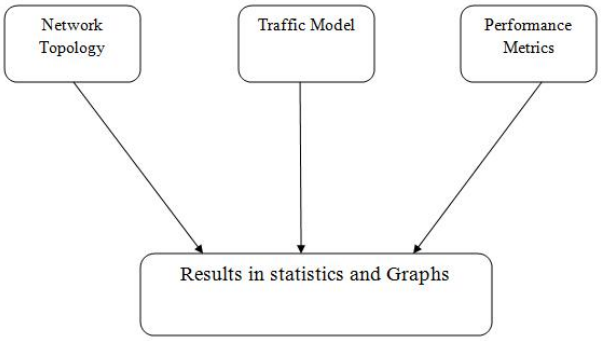


Figure 3: Architecture of our tools

**V .1 Network topology**

To study our TCP and ICN behavior we built our Network and application model shown in figure (4), in which source nodes and sink nodes connect to router 1 or router 2. The bandwidth between the two routers is much lower than the other links, which causes the link between the routers to be a bottleneck. (Traffic can be either uni-directional or bidirectional).

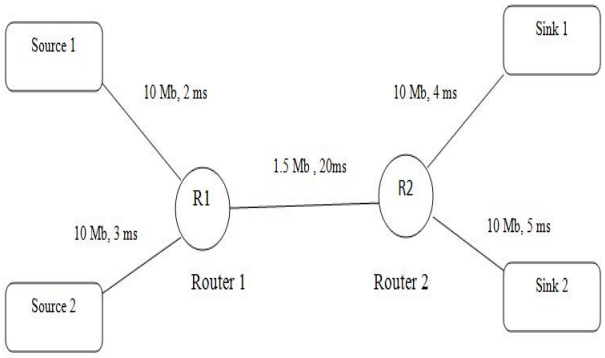


Figure 4: Network topology

**V.2 Traffic Model**

The tool attempts to apply the typical traffic settings. In our application include the FTP traffic that uses infinite, non-stop file transmission, which begins at a random time and runs on the top of TCP. Implementation details and a comparative analysis of TCP Tahoe, Reno, New-Reno, SACK and Vegas choices of TCP variant are decided by users.

**V.3 Performance evaluation metrics**

The metrics used in our simulations are Throughput and Drop ratio. Throughput is the total elapsed flow since the beginning of simulation time. Throughput may also includes retransmitted traffic (repeated packets).Drop ratio is the total rate of packet loss during the simulation time. To obtain network statistics, we measure also the drop ratio metric that result in the failure of the receiver to decode the packet and simulation time is 100 seconds.

Robust TCP is poor in performance as a standalone TCP but after adding the ICN it becomes much better (see figure 5) and accurate than TCP New-Reno as show in the figure (6). To evaluate our scenario, we compare TCP Robust with other TCP variants (TCP New-Reno) by using different metrics that will show us clearly the improvement of our TCP version compared to others. (Figure 6 and 7).

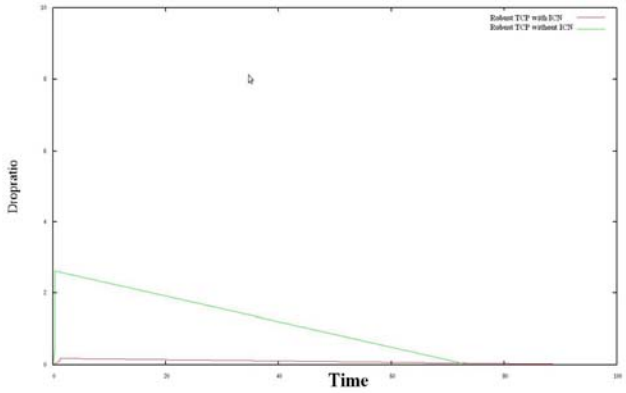


Figure 5: Comparison between TCP Robust before and after adding ICN algorithm.

The main difference between Robust and New-Reno TCP occurs in the reaction of each protocol. In the TCP New-Reno the reaction will be whenever an error or congestion occurs on the network by slowdown the transmission without being accurate if there is a congestion or not. In addition of that the main problem of New-Reno TCP that it suffers from the fact that it takes one RTT to detect each packet loss. When the ACK for the first retransmitted segment is received only then we can deduce which other segment was lost. This problem of inaccuracy in TCP New-Reno is solved by the ICN algorithm that the ICN receive the packet and check the presence of congestion by using the normal and congestion phase and by adding the timestamp option which can be make sure of the presence of congestion or no. The deduction of congestion in TCP Robust is different from New-Reno, it will be deduced after signaling ECN from ICN to TCP robust, and then the TCP reacts by decreasing the transmission. This accuracy in detection of congestion can be up to 24 % as difference

between the two protocols (Figure 6) before reaction of each one and starting slowdown retransmission.
Due the fast reaction of TCP robust, the transmission of TCP become less than in TCP New-Reno which means that the throughput in the TCP robust must be less than in New-Reno, this is clear and deduced in the figure 7.

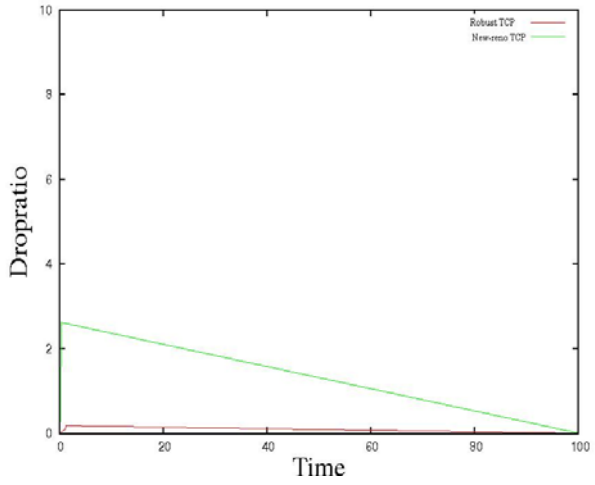


Figure 6: Comparison between TCP Robust and TCP New-Reno

In figure (6) represents that the drop ratio is less in Robust than in New-Reno due that TCP reacts only when receiving ECN which make its reaction faster.

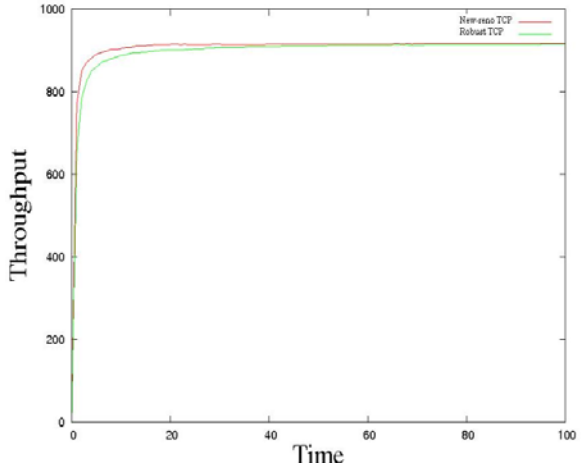


Figure 7: Comparison between TCP Robust and TCP New-Reno

In figure (7) Robust TCP algorithm reaction is faster than the Reaction of New-Reno, thus Throughput in New-Reno is higher than when using Robust TCP. Congestion detection used by ICN algorithm is more accurate when using the timestamp option for detecting a spurious timeout which improve more the performance of TCP.
The main difference between spurious timeout algorithms relies on the method how to detect spurious timeout by solving the retransmission ambiguity in many circumstances. After clarifying this ambiguity TCP can tell whether the data is there is spurious timeout has happened or not. DSACK, F-RTO and Robust TCP can see the problem of spurious timeout in different aspects.
DSACK, an extension of TCP SACK, works it out in the sequence space. It requires the TCP receiver explicitly acknowledging duplicate segments with duplicate SACK options. F-RTO algorithm is used for detecting spurious retransmission timeouts with TCP. It is a TCP sender-only algorithm that does not require any TCP options to operate-RTO delays the decision of loss recovery and waits further two ACK. If the first arrived ACK forwards the sender's transmitting window, TCP concludes a spurious timeout and resume transmitting new data.
Our approach is different than other TCP by using an algorithm of congestion detection outside the TCP code, where it can detect congestion and spurious timeout by using the timestamp option at the occurrence of loss or congestion event. The main advantage of ICN with timestamp algorithm is that it can work with spurious timeouts and the others loss events by detecting the congestion in the network immediately and then directly will be notified to Robust TCP in order that TCP after this action will reduce its window, which can improve very well the performance of our TCP.

## VI. CONCLUSION AND FUTURE WORK

This paper has proposed a new algorithm, which is implemented as a stand alone component and not inside a TCP stack. This algorithm that interacts with a classic version of TCP is able to detect congestion and notify directly the loss to the Robust TCP through the congestion notification (ECN) in order to reduce its window which leads to a Robust TCP compared to other variants like New-Reno and SACK TCP. In our work we demonstrate that congestion event detection can be realized independently of the TCP code in sake of better detecting congestion occurring in the network.
Following this work and the results obtained so far, we are currently planning to develop more the detection of congestion by using the delay-based

in the congestion detection algorithm (ICN) and the effect of fast reaction of TCP robust in the Network.

## REFERENCES


[1] P.sarolahti and M. kojo. Forward rto-recovery (f-tro): An algorithm for detecting spurious retransmission timeouts with tcp and the stream control transmission protocol (sctp).rfc 4138,IETF, August 2005.

[2] P. Anelli and F. Harivelo, E. lochin. On TCP congestion events detection.

[3] Reiner Ludwig and Randy H..Katz. The Eifel algorithm: making TCP robust against spurious retransmissions. SIGCOMM Comput. Common. Rev., 30(1):30-36, 2000.

[4] K. Ramakrishnan, S. Floyd, and D. Black. The addition to explicit congestion notification (ECN) to ip. Request for comments 3168,IETF, September 2001.

[5] Bhandarkar, S. and Reddy, A.L.N. (2004), Networking, May TCP-Dcr: Making TCP Robust to Non-Congestion Events.

[6] S. Floyd ICSI December 2003. RFC 3649 - High Speed TCP for Large Congestion Windows.

[7] A Comparative Analysis of TCP Tahoe, Reno, New- Reno, SACK and Vegas.

[8] RFC 793 Transmission Control Protocol, September 1981.

[9] RFC 3649 High Speed TCP for Large Congestion Windows, S. Floyd December 2003.


## BIOGRAPHY:

**Dr. Seifedine Kadry** received his master's degree in Modeling and Intensify Calculus (2001) from the Lebanese University – EPFL - INRIA. He received his PhD (2003-2007) in applied mathematics from Blaise Pascal University, and IFMA, France. He worked as Head of Software Support and Analysis Unit at First National Bank. He published many papers in national and international journals. His research interests are in the areas of Computer Science, Numerical Analysis and Stochastic Mechanics. Since 2009, Dr Kadry is an associate professor at the Lebanese University.

**Dr. Mohamed Smaili** received the masters' degree in Computer Science from the University of Monpelier II (USTL), France, in 1990. He received his Ph.D. in Computer Science from the University of Monpelier II (USTL), France, in 1993. Since 1994 he teaches at the Lebanese University, Faculty of Science. He published many papers in national and international conferences. His research interests are fuzzy logic, digital circuits and simulation.

**Issa Kamar** received his master's degree in Computer Science and communication (2004) from the AUL University – LEBANON. Currently He is a his PhD student at the Lebanese University. The goal of his PhD thesis is to study and improve the TCP protocol.