Multiple Valued Logic (MVL) Reduction Operator, its Synthesis and Application on Network Congestion

Adib Kabir Chowdhury, Muhammad Ibrahim, Veeramani Shanmugam and Ashutosh Kumar Singh

Department of Electrical and Computer Engineering, Curtin University, Sarawak Malaysia, CDT 250, 98009 Miri, Sarawak, Malaysia

Abstract. In this paper, reduction logical operator (RLO) has been proposed to control bottleneck link size by realizing Multi-Valued Logic (MVL) function. A novel Switch Delay Link Mechanism (SDLM) Algorithm using Transmission Control Protocol (TCP) has been represented to serve as a basic input generator for base and reduced link size of the network bottleneck, which gives input to the RLO. On the other hand, the SDLM algorithm produces base and reduced latency, which is later fed to the RLO for better performance. A complete RLO function can be further synthesized to obtain faster and efficient data processing. SDLM algorithm merges with RLO operator to overcome the network congestion problem of fixed networks. The advantages of RLO-SDLM algorithm is demonstrated with packet drop in fixed networks. The comparison against RLO-SDLM algorithm is based on the usage of network bottleneck link size and the delay to determine the sending rate and packet loss. Overall the algorithm shows an average improvement of 0.52% compared to the other existing algorithm. We used NS2 software simulator to experiment different set of flows to achieve an optimum result.

Keywords: Switch Delay Link Algorithm, MVL, synthesis, Network Congestion, Reduction Logic Operator, TCP.
PACS: 89.20.Ff

INTRODUCTION

Network congestion has lately attract attention in the past few decades due to the significant increased amount of internet users, which course inefficient utilization of resources. Congestion is said to occur when the amount of resource demands exceed the capacity of the network and causes loss of packets and link failure. It has been observed that congestion mainly occurs due to the limited capacity of the network bottleneck. In this era of technology today, communication network has become the fundamental key of modern technology. The scope of communication has significantly increased in the past few decades. This increment in communications would not be possible without the progressive advancement of communications network. TCP has become one of the major network protocols on which efficient methods are established to reduce information loss. Also it has been designed to dynamically adjust the window size of competing connections so as to utilize network resources in the most efficient way.

Multiple Valued Logic (MVL) system has been used in the design of various logic and arithmetic systems, which also includes few special purpose digital processors, circuits of next generation quantum computers, signal encoding in long distant communication, efficient memory design, synthesis of combination and sequential circuits and many other in field of digital communications. Several algorithms have been proposed in the literature for network congestion. Existing algorithm requires much computational complexity and higher link size/ latency. Hence it takes extensive amount of time to compute the optimal reduction in size and latency. Most of the existing efficient algorithms are computationally complex and produce unreduced size and latency of the bottleneck.

MVL RLO has also an important role in the betterment of network congestion. The only draw-back of MVL RLO is that it needs an external algorithm to compute the series of reduced link size and latency. RLO gives the optimal result by using its logic operation. Existing methods are not efficient enough to compete against the recent Additive increase/ Multiplicative decrease (AIMD) algorithms.

In this paper, a novel MVL reduction operator defined and explained. Combining two separate window literal a minterm has been formed [1]. Some basic MVL operators have been also revised in order to help explain RLO. SDLM has been represented to produce series of base, reduced link size and base, reduced latency for a fixed network. Each of these base forms the row and reduced link size/ latency forms the column of the RLO. The RLO decides the reduction of the link size/ latency based on operator logic. MVL functions at an optimal level. It utilizes these postulate and logical operators for the synthesis

and realization of MVL functions. The advantages of RLO-SDLM algorithm is shown by comparing it with latest AIMD algorithms. Moreover the amount of information received from sender to receiver has increased by an average of 0.53% which reasons the feasibility of the experiment.

To the extent of literature done, the proposed RLO-SDLM method shows that the information receiving rate at the receiver terminal has drastically improved. The proposed method is a hybrid of SDLM algorithm and MVL system which is a novel approach to generate an efficient rate of information received with reduced link size and delay, compared to other existing algorithms or rules.

LITERATURE REVIEW

TCP [2-4] is mostly used end-to-end transport layer protocol in the internet, which account of the majority of internet data traffic. Generally, TCP uses a form of end-to-end flow control. In TCP, when a sender sends a packet, the receiver acknowledges receipt of the packet. A sending source can use the acknowledgement arrival rate as a measure of network congestion. When it successfully receives an acknowledgment, a sender knows that the packet reached its destination. The sender can then send new packets on the network. Both the sender and the receiver agree on a common window size for packet flow [5]. The window size represents the number of bytes that the source can send at a time. The window size varies according to the condition of traffic in the network to avoid congestion [6]. Over few decade of development, according to the constant demand for network transmission, the researchers also based on this have proposed a series of improved algorithm, such as: TCP Reno [7], [5], TCP SACK [8], [3] and TCP Vegas [3].

In TCP Reno, the fast retransmit was modified to operate with fast recovery; this new algorithm prevents the communication channel from going empty after Fast Retransmit, thereby avoiding the need to Slow-Start to re-fill it after a single packet loss [7]. Fast Recovery operates by assuming each dup ACK received represents a single packet having left the pipe. Thus, during Fast Recovery the TCP sender is able to make intelligent estimates of the amount of outstanding data. A TCP sender enters fast Recovery after receiving an initial threshold of dup ACKs. Once the threshold (generally is 3) of dup ACKs is received, the sender retransmits one packet and reduces its congestion window by one half. After entering Fast Recovery and retransmit a single packet, the sender effectively waits until half of a window of dup ACKs have been received, and then sends a new packet for each additional dup ACK that is received. Upon receipt of an ACK for new data, the sender exits Fast Recovery [8]. Reno significantly improves upon the behaviour of Tahoe TCP when a single packet is dropped from a window of data, but can suffer from performance problems when multiple packets are dropped from a window of data.

TCP SACK was a conservative extension of Reno's congestion control, in that they use the same algorithms for increasing and decreasing the congestion window, and make minimal changes to the other congestion control algorithms. Adding SACK (Selective Acknowledgement) to TCP does not change the basic underlying congestion control algorithms. The SACK TCP implementation preserves the properties of Tahoe and Reno TCP of being robust in the presence of out-of-order packets, and uses retransmit timeouts as the recovery method of last resort [9]. The main difference between the SACK TCP implementation and the Reno TCP implementation is in the behaviour when multiple packets are dropped from one window of data. During Fast Recovery, SACK maintains a variable called pipe that represents the estimated number of packets outstanding in the path. The sender only retransmits data when estimated number of packets in the path is less than the congestion window.

TCP Vegas accepts more sophisticated bandwidth estimation scheme. It uses the difference between expected and actual flow rates to estimate the available bandwidth in the network. The idea is that when the network is not congested, the actual flow rate will be close to the expected flow rate. Otherwise, the actual flow rate will be smaller than the expected flow rate [2]. TCP Vegas, using this difference in flow rates, estimates the congestion level in the network and updates the window size accordingly. This difference in the flow rates can be easily translated into the difference between the window size and the number of acknowledged packets during the round trip time.

DESCRIPTION OF EXISTING MVL OPERATORS AND ALGEBRA:

Consider an *m*-valued *y*-variable function f(x), where $x = \{x_1, x_2, \cdots, x_m\}$ and x_i takes on values from $R = \{0,1,2,\cdots, y-1\}$, where "*y*" is the radix. The function f(x) is a mapping $f : R^{y} \Rightarrow R$. There are y^{y^m} different possible functions. In this paper for the experiment purpose a maximum of *n* valued 2 variable functions are investigated. The value of *n* depends upon the SDLM algorithm.

Definition 3.1- A min (minimum) operator is defined as below, where a_n and b_n is a definite value and both of them are a member of set of all real values R [10],[11],[12], [13], [14-16] ,[17] and [18]:

$$a_1, a_2, \cdots, a_n \in R\,,\, b_1, b_2, \cdots, b_n \in R \text{ and}$$
$$0 \le \{a, b\} \le (y-1) \quad (1)$$

$$\min(a_1, a_2, \cdots, a_n) = a_1 \sqcap a_2 \sqcap \cdots \sqcap a_n \text{ and}$$
$$\min(b_1, b_2, \cdots, b_n) = b_1 \sqcap b_2 \sqcap \cdots \sqcap b_n \quad (2)$$

Therefore,

$$\min(a, b) = a \sqcap b = a \cap b \quad (3)$$

And the minimum operator could be represented by the following relation [14-16] and [17]:

$$\min(a, b) = \begin{cases} a & if \quad a < b \\ b & otherwise \end{cases} \quad (4)$$

Definition 3.2- A max (maximum) operator is defined as below [10],[11],[12], [13], [14-16] ,[17] and [18]:

$$a_1, a_2, \cdots, a_n \in R\,,\, b_1, b_2, \cdots, b_n \in R \text{ and}$$
$$0 \le \{a, b\} \le (y-1) \quad (5)$$

$$\max(a_1, a_2, \cdots, a_n) = a_1 + a_2 + \cdots + a_n \text{ and}$$
$$\max(b_1, b_2, \cdots, b_n) = b_1 + b_2 + \cdots + b_n \quad (6)$$

$$\text{Therefore } \max(a, b) = a + b = a \cup b \quad (7)$$

And the maximum operator could be represented by the following relation:

$$\max(a, b) = \begin{cases} a & if \quad a > b \\ b & otherwise \end{cases} \quad (8)$$

Definition 3.3- The Complement of x is defined as below, where x a member of set of all real values R [10],[11],[12], [13], [14-16] ,[17] and [18]:

$$x \in R\,,\, a_1, a_2, \cdots, a_n \in R \text{ and } b_1, b_2, \cdots, b_n \in R$$
$$0 \le \{a, b\} \le (y-1) \text{ and } a \le x \le b \quad (9)$$

Therefore, complement of x can be represented as,

$$\bar{x} = y - 1 - x$$

Definition 3.4- According to [1] the window literal or short literal is defined as

$${}^{a}x^{\,b} = \begin{cases} b & if \quad x = a \\ 0 & otherwise \end{cases} \quad (10)$$

where, $a_1, a_2, \cdots, a_n \in R\,,\, b_1, b_2, \cdots, b_n \in R$ and

$$0 \le \{a, b\} \le (y-1) \quad (11)$$

b is called the value of the literal. Considering the definition of window literal provides the opportunity of eliminating the usage of *complementary literal (CL)*. Thus in this paper *complementary-literal* is not extensively used during the synthesis of MVL functions.

PROPOSED METHODOLOGY

A set of novel MVL operators and SDLM algorithm has been proposed to reduce link and latency of network bottleneck. The functioning of logic operators and SDLM algorithm has been shown to understand the process of reduction. RLO utilizes SDLM algorithm to retrieve the base and reduced link/latency. The n valued 2 variable RLO then decides on which is the most reduced link/latency for the bottleneck. In the following segment, each of the proposed sub-section will be presented.

Definition 3.4- A reduction operator is defined as below, where a_n and b_n is a definite value and both of them are a member of set of all real values R:

$$a_1, a_2, \cdots, a_n \in R\,,\, b_1, b_2, \cdots, b_n \in R \text{ and}$$
$$0 \le \{a, b\} \le (y-1) \quad (12)$$

And the reduction operator could be represented by the following relation:

$$\text{Re}\,duction(a, b) = \begin{cases} a & if \quad b \ge a \\ 0 & otherwise \end{cases} \quad (13)$$

The radix y depends upon the step size of base link and decrement ratio generated by SDLM algorithm. Below is an example of the RLO function for fixed link sizes/ delay of the bottleneck,

TABLE 1. An example of MVL Reduction Function as b being Reduced link size and a being the base link size of the bottleneck

MVL Reduction Operator	***b*** **0.250**	**0.625**	**1.125**	**1.625**	**2.000**
0.250	0.250	0	0	0	0
0.625	0.250	0.625	0	0	0
1.125 ***a***	0.250	0.625	1.125	0	0
1.625	0.250	0.625	1.125	1.625	0
2.000	0.250	0.625	1.125	1.625	2.000

The reduction function can be expressed by f. Each of the columns can be synthesized separately [1]. Different columns can be expressed by $f_{1.....n}$. According to [1] the synthesis of the reduction function can be expressed as following. The first, second, third, fourth and fifth column and its synthesized representation are shown below,

$$f_1 = {}^0x_1{}^0 \square {}^0x_2{}^0 + {}^0x_1{}^0 \square {}^1x_2{}^0 + {}^0x_1{}^0 \square {}^2x_2{}^0 + {}^0x_1{}^0 \square {}^3x_2{}^0 + {}^0x_1{}^0 \square {}^4x_2{}^0 \text{ and}$$
$$f_1 = {}^0x_1{}^0 \square \left({}^0x_2{}^0 + {}^1x_2{}^0 + {}^2x_2{}^0 + {}^3x_2{}^0 + {}^4x_2{}^0 \right) \quad (14)$$

$$f_2 = {}^1x_1{}^1 \square {}^1x_2{}^1 + {}^1x_1{}^1 \square {}^2x_2{}^1 + {}^1x_1{}^1 \square {}^3x_2{}^1 + {}^1x_1{}^1 \square {}^4x_2{}^1 \text{ and}$$
$$f_2 = {}^1x_1{}^1 \square \left({}^1x_2{}^1 + {}^2x_2{}^1 + {}^3x_2{}^1 + {}^4x_2{}^1 \right) \quad (15)$$

$$f_3 = {}^2x_1{}^2 \square {}^2x_2{}^2 + {}^2x_1{}^2 \square {}^3x_2{}^2 + {}^2x_1{}^2 \square {}^4x_2{}^2 \text{ and}$$
$$f_3 = {}^2x_1{}^2 \square \left({}^2x_2{}^2 + {}^3x_2{}^2 + {}^4x_2{}^2 \right) \quad (16)$$

$$f_4 = {}^3x_1{}^3 \square {}^3x_2{}^3 + {}^3x_1{}^3 \square {}^4x_2{}^3 \text{ and } f_4 = {}^3x_1{}^3 \square \left({}^3x_2{}^3 + {}^4x_2{}^3 \right) \quad (17)$$

$$f_5 = {}^4x_1{}^4 \square {}^4x_2{}^4 \quad (18)$$

The synthesized representation of the entire MVL reduction function can be represented as the following,

$$\begin{aligned} f = {}& {}^0x_1{}^0 \square \left({}^0x_2{}^0 + {}^1x_2{}^0 + {}^2x_2{}^0 + {}^3x_2{}^0 + {}^4x_2{}^0 \right) \\ & + {}^1x_1{}^1 \square \left({}^1x_2{}^1 + {}^2x_2{}^1 + {}^3x_2{}^1 + {}^4x_2{}^1 \right) \\ & + {}^2x_1{}^2 \square \left({}^2x_2{}^2 + {}^3x_2{}^2 + {}^4x_2{}^2 \right) \\ & + {}^3x_1{}^3 \square \left({}^3x_2{}^3 + {}^4x_2{}^3 \right) + {}^4x_1{}^4 \square {}^4x_2{}^4 \end{aligned} \quad (19)$$

The representation shows that the synthesized expression requires only four min gates to represent the function. The structure of the MVL RLO operator can be shown in a matrix form as represented below,

Reduction Operator	a	b	c	$\cdots$	nth
a	a	0	0	$\cdots$	0
b	a	b	0	$\cdots$	0
c	a	b	c	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
nth	a	b	c	$\cdots$	nth

Figure 1. The structure of MVL RLO operator

RLO-SDLM (SWITCH DELAY-LINK MECHANISM) ALGORITHM:

Based on the RLO-SDLM a pseudo code has been prepared. The main frame pseudo code is divided into three procedures. Each of the procedure generates output to serve as the input to the next procedure. The step size of the bottleneck link size/ delay and their values are fed to the final main procedure *mvl_Reduction_Operator()*. Below the entire pseudo code for the algorithm is represented according to the appearance of their activity,

Procedure *generateBaseLinkSize()*;

initialize determinantCount = 0;
senderCounter=0;
for each sender $L_s : s < n$
evaluate each flow determinant
$f_{D_L} = evalDeterminantLinkSize(L_s)$;
if ($f_{D_L} > 0$)
determinantCount = determinantCount + f_{D_L} ;
$track[track\_Count] = \det er \min antCount$;
senderCounter++;
else
$pr \text{int } ErrMsg("No \quad Sender \quad Found")$;

$$Bottleneck\_BaseLS = \frac{\sum_{i=0}^{n} \left(\frac{track\_Count[i]-1}{2} \right)}{senderCounter};$$

return *Bottleneck_BaseLS ;*
end

Figure 2. Procedure which generates the base link size of the bottleneck.

.

Procedure *bottleneckLinkDecrementRatio(int reductionPower, double* *Bottleneck_BaseLS* ***);***

Calculate B_{LDR} **to generate decrement ratio:**

$$B_{LDR} = \frac{Bottleneck\_BaseLS}{2^{reductionPower+1}};$$

return B_{LDR} **;**
end

Figure 3. Procedure which generates the decrement ratio (DR).

```
Procedure bottleneckStepLinkSizes(int redPower,
double Bottle_BaseLS );
    dec_ratio, step_Count, base_Link = 0;
    dec_Ratio=decbottleneckLinkDecrementRatio
    ( redPower, Bottle_BaseLS );
    do
        step_Size = base_Link / dec_Ratio;
        Step_size : step_Count < step_Size
        link_Steps[step_Count] = f_{D_{L_n}} - B_{LDR};
        P_d = packetDrop( f_{D_{L_n}} );
        if ( f_{D_{L_n}} - B_{LDR} ≥ 0 ) exit loop;
       else continue;
    while ( P_d != 0 )
return link_Steps[step_Count];
end
```

Figure 4. Procedure which generates the step link sizes of the bottleneck

MVL RLO to Retrieve Link/Delay of Bottleneck

The MVL RLO takes input from the SDLM algorithm and outputs the final link size and delay of the bottleneck. The mechanism is represented as a pseudo code as shown below,

```
Procedure mvl_Reduction_Operator();
Initialize N_L = senders; step_Count = 0;
mvl_Reduction_Operator[]; rPower = [3 ≤ n ≤ N_L];
BaseLS = generateBaseLinkSize();
Variable = bottleneckStepLinkSizes(int rPower,
double BaseLS );
    for each Step_size : size != null
    base_Size = link_Steps[step_Count];
    reduced_Size = link_Steps[step_Count];
    if ( base_Size ≥ reduced_Size )
        reduction = reduced_Size;
    else
        reduction = 0;
        mvl_Reduction_Operator[step_Count];
        step_Count++;
end
```

Figure 5. Procedure of the RLO which reduces the link size/ latency of the bottleneck

MATHEMATICAL BACKGROUND OF REDUCTION

The SDLM algorithm emphasizes on evaluating the series of steps of reduced link size and latency of the bottleneck. In order to obtain the steps a thorough calculation for each parameter has to be done. Below both of the calculation are shown,

Bottleneck Link Size Determinant and Its Calculation

Information flow of the network depends on the bottleneck. The information which flows through the bottleneck depends upon its delay and link size. Each flow (sender) of the network determines the total link size and delay of the bottleneck. Each flow has its own determinant based on its link size and delay. Each flow determinant f_{D_L} can be given by the equation below:

$$f_{D_L} = \frac{d_a \left[L_s - 1 \right]}{2 d_a} \tag{20}$$

d_a represents the sequence of delay for each of the flow (sender) of the network, where $d_a = \{d_0, d_1, d_2 \cdots d_n\}$. L_s represents the sequence of link size for each of the flow (sender) of the network, where $L_s = \{L_0, L_1, L_2 \cdots L_n\}$. There exists n^{th} determinant of n^{th} flow (sender), therefore n^{th} determinant for the link size of bottleneck can be given by,

$$f_{D_{L_n}} = f_{D_{L_0 \cdots L_n}} = \frac{1}{\sum_{i=0}^{n} N_{L_i}} \left[\frac{d_{a_0} \left[L_0 - 1 \right]}{d_{a_0} 2} + \frac{d_{a_1} \left[L_1 - 1 \right]}{d_{a_1} 2} + \frac{d_{a_2} \left[L_2 - 1 \right]}{d_{a_2} 2} + \cdots + \frac{d_{a_n} \left[L_n - 1 \right]}{d_{a_n} 2} \right] \tag{21}$$

N_L represents the number of flows (senders). $f_{D_{L_n}}$ determines the base link size of the bottleneck. This link size could be further reduced to optimize the efficiency of the information flow. Further reduction of the link size is determined by the Bottleneck Link Decrement Ratio B_{LDR}. B_{LDR} can be obtained by the equation below,

$$B_{LDR} = \frac{1}{2^{n+1} \sum_{i=0}^{n} N_{L_i}} \left[\frac{d_{a_0} \left[L_0 - 1 \right]}{d_{a_0} 2} + \frac{d_{a_1} \left[L_1 - 1 \right]}{d_{a_1} 2} + \frac{d_{a_2} \left[L_2 - 1 \right]}{d_{a_2} 2} + \ldots + \frac{d_{a_n} \left[L_n - 1 \right]}{d_{a_n} 2} \right] \quad (22)$$

Where $2^{n+1} = 2 2^n$ determines the decrement ratio and $n \in R$ and $3 \le n \le N_L$, where n is the number of senders. The final decrement ratio can be represented with the equation below,

$$B_{LDR} = \frac{f_{D_{L_n}}}{2^{n+1}} \quad (23)$$

The following equation can be used to determine the number of senders of the network,

$$n = \log_2^{\left[\frac{f_{D_{L_n}}}{B_{LDR}} \right]} - 1 \quad (24)$$

Packet drop P_d can be determined by the equation which is represented as below. P_d is represented in such a way that, $P_d \in R$ and $P_d \ge 0$. P_{s_i} represents the sequence of packet sent within the time span of T_s where $P_{s_i} = \left\{ P_{s_0}, P_{s_1}, P_{s_2} \cdots P_{s_n} \right\}$. P_r represents the sequence of packet received within the time span of T_r, where $P_{r_i} = \left\{ P_{r_0}, P_{r_1}, P_{r_2} \cdots P_{r_n} \right\}$.The equation below represents the equation for the packet drop P_d. The packet drop is calculated in percentage with respect to the total packet sent by sender and total packet received by the receiver.

$$P_d = \frac{T_r \sum_{i=0}^{n} \Pr_i}{T_s \sum_{i=0}^{n} \mathrm{P}s_i} \times 100\%; where [T_S = T_r] \quad (25)$$

In order to achieve the optimal bottleneck link, the n[th] determinant for the link size has to be reduced by the bottleneck link decrement ratio B_{LDR}. The reduction will reach up until a certain limit where by the packet drop will get approximately close to zero as shown below:

$$f_{D_{L_n}} - B_{LDR} \ge 0 \text{ and } P_d \approx 0 \quad (26)$$

Bottleneck Delay Determinant and Its Calculation

As for the information flow of the network depends on the delay of the bottleneck, it is necessary to consider the delay calculation. The information which flows through the bottleneck depends upon its delay. Each sender of the network determines the total delay and of the bottleneck. Each sender has its own determinant based on its delay. Each flow determinant f_{D_L} can be given by the equation below:

$$f_{D_L} = \frac{L_s \left[d_a - 1 \right]}{2 L_s} \quad (27)$$

d_a represents the sequence of delay for each of the sender of the network, where $d_a = \left\{ d_0, d_1, d_2 \cdots d_n \right\}$. L_s represents the sequence of link size for each of the sender of the network, where $L_s = \left\{ L_0, L_1, L_2 \cdots L_n \right\}$. There exists n[th] determinant of n[th] sender, therefore n[th] determinant for the delay of the bottleneck can be given by,

$$f_{D_{d_{a_n}}} = f_{D_{d_{a_0} \cdots d_{a_n}}} = \frac{1}{2 \sum_{i=0}^{n} N_{L_i}} \left[\frac{L_0 \left[d_{a_0} - 1 \right]}{L_0} + \frac{L_1 \left[d_{a_1} - 1 \right]}{L_1} + \frac{L_2 \left[d_{a_2} - 1 \right]}{L_2} + \ldots + \frac{L_n \left[d_{a_n} - 1 \right]}{L_n} \right] \quad (28)$$

N_L represents the number of senders. $f_{D_{d_n}}$ determines the base delay of the bottleneck. This delay could be further reduced to optimize the efficiency of the information flow. Further reduction of the delay is determined by the Bottleneck Delay Decrement Ratio B_{DDR}. B_{DDR} can be obtained by the equation below,

$$B_{DDR} = \frac{1}{2^{n+2} \sum_{i=0}^{n} N_{L_i}} \left[\frac{L_0 \left[d_{a_0} - 1 \right]}{L_0} + \frac{L_1 \left[d_{a_1} - 1 \right]}{L_1} + \frac{L_2 \left[d_{a_2} - 1 \right]}{L_2} + \ldots + \frac{L_n \left[d_{a_n} - 1 \right]}{L_n} \right] \quad (29)$$

Where $2^{n+1} = 2 2^n$ determines the decrement ratio and $n \in R$ and $3 \le n \le N_L$, where n is the number of senders. The final decrement ratio can be represented with the equation below,

$$B_{DDR} = \frac{f_{D_{d_n}}}{2^{n+1}} \quad (30)$$

The following equation can be used to determine the number of senders of the network,

$$n = \log_2^{\left\lceil \frac{f_{D_{d_n}}}{B_{DDR}} \right\rceil} - 1 \quad (31)$$

Packet drop P_d can be determined by the equation which is represented as below. P_d is represented in such a way that, $P_d \in R$ and $P_d \geq 0$. P_{s_i} represents the sequence of packet sent within the time span of T_s where $P_{s_i} = \{P_{s_0}, P_{s_1}, P_{s_2} \cdots P_{s_n}\}$. P_r represents the sequence of packet received within the time span of T_r, where $P_{r_i} = \{P_{r_0}, P_{r_1}, P_{r_2} \cdots P_{r_n}\}$.The equation below represents the equation for the packet drop P_d. The packet drop is calculated in percentage with respect to the total packet sent by sender and total packet received by the receiver.

$$P_d = \frac{T_r \sum_{i=0}^{n} \Pr_i}{T_s \sum_{i=0}^{n} \mathrm{P}\, s_i} \times 100\%; where [T_S = T_r] \quad (32)$$

In order to achieve the optimal bottleneck delay, the n[th] determinant for the delay has to be reduced by the bottleneck delay decrement ratio B_{DDR}. The reduction will reach up until a certain limit where by the packet drop will get approximately close to zero as shown below:

$$f_{D_{d_n}} - B_{DDR} \geq 0 \text{ and } P_d \approx 0 \quad (33)$$

SIMULATION RESULTS AND DISCUSSION

In order to evaluate the novel RLO-SDLM algorithm experiments have been conducted based on the NS2 software simulation. The simulation results obtained from NS2 not only helps to understand the behaviour of the RLO-SDLM algorithm but also it helps to understand the effect of bottleneck with increased number of network users. The figure 6 below shows a simple topology which is used in the simulation. The topology is a bum-bell topology network. The link size of the bottleneck is set to 5 Mbps. The link that connects the senders and the receivers to the router has 5 Mbps bandwidth. The latency of the bottleneck is set to 30 ms (millisecond). The router queue is set to 0.5 queuepos in NS2 software.

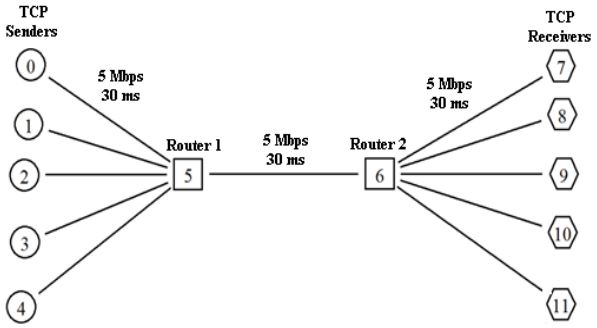

Figure 6. The dun-bell topology network used as a testbench for the RLO-SDLM algorithm

The figure below shows the reduced link size and latency of the bottleneck for a five flow network. The throughput from the network is calculated over a period of 300 seconds. Calculations and simulation results have shown that using this topology the throughput receiving rate becomes 99.96%. The simulation result also suggests that the improvement achieved over algorithm from [19] has increased by 0.59%.

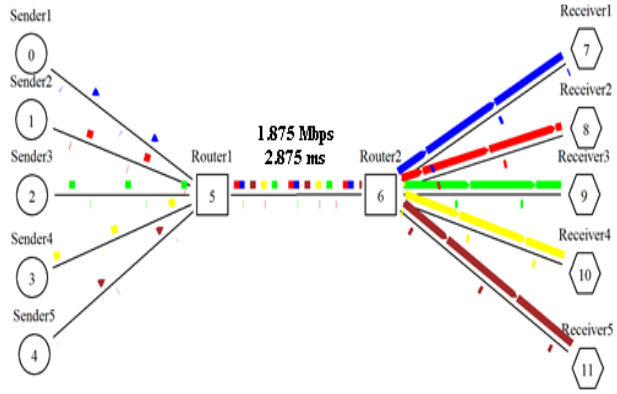

Figure 7. The dun-bell topology network used as a testbench for the RLO-SDLM algorithm

Figure 8 shows that at link size 2.000 Mbps with latency 2.875 ms the packet does not drop due to the balance of adequate amount of information respective to the bottleneck capacity. Figure 9 shows that at link size 1.875 Mbps and delay of 2.875 ms the packet starts dropping due to the flow of information which exceeds the bottleneck capacity. Figure 10 shows that at link size 1.750 Mbps and delay of 2.875 ms the packet starts dropping with higher frequency due to the heavy flow of information which exceeds the bottleneck capacity.

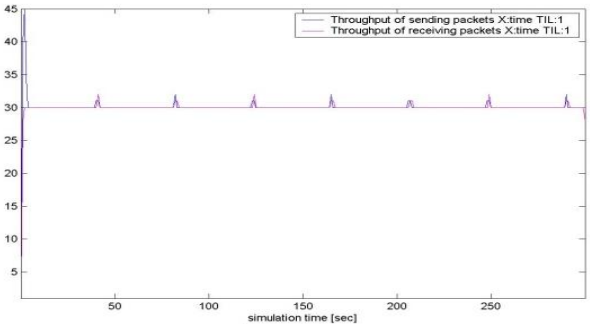

Figure 8. Packet stops dropping at reduced link size 2.000 Mbps with latency 2.875 ms for a five flow network.

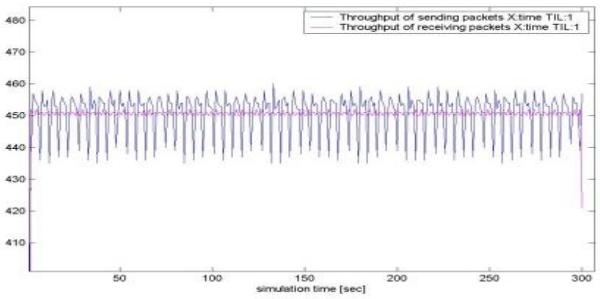

Figure 9. Packet starts dropping at reduced link size 1.875 Mbps with latency 2.875 ms for a five flow network.

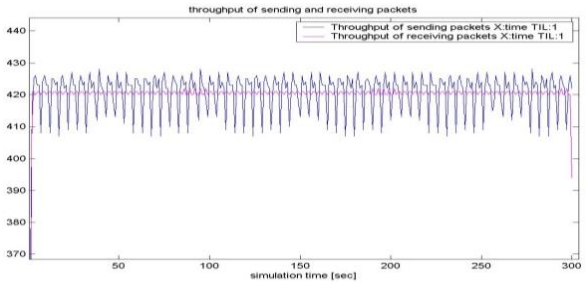

Figure 10. Frequency of packet drop increases at reduced link size 1.750 Mbps with latency 2.875 ms for a five flow network

The figure below shows the reduced link size and latency of the bottleneck for a four flow network. The throughput from the network is calculated over a period of 300 seconds. Calculations and simulation results have shown that using this topology the throughput receiving rate becomes 99.94%. The simulation result also suggests that the improvement achieved over algorithm from [19] has increased by 0.21%.

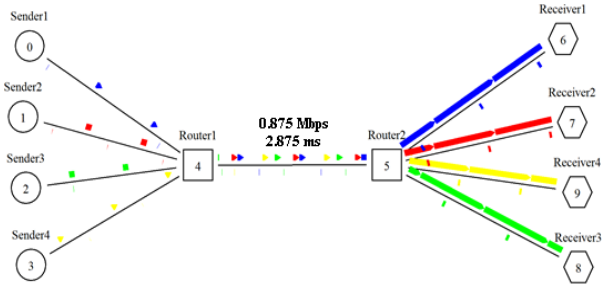

Figure 11. Reduced bottleneck link size and delay for four flow fixed network.

Figure 12 shows that at link size 1.000 Mbps with latency 2.875 ms the packet does not drop due to the balance of adequate amount of information respective to the bottleneck capacity. Figure 13 shows that at link size 0.875 Mbps and delay of 2.875 ms the packet starts dropping due to the flow of information which exceeds the bottleneck capacity. Figure 14 shows that at link size 0.750 Mbps and delay of 2.875 ms the packet starts dropping with higher frequency due to the heavy flow of information which exceeds the bottleneck capacity.

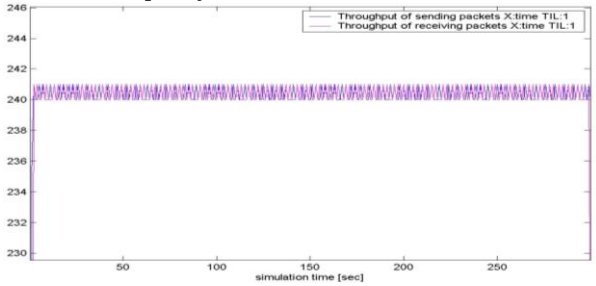

Figure 12. Packet stops dropping at reduced link size 1.000 Mbps with latency 2.875 ms for a four flow network.

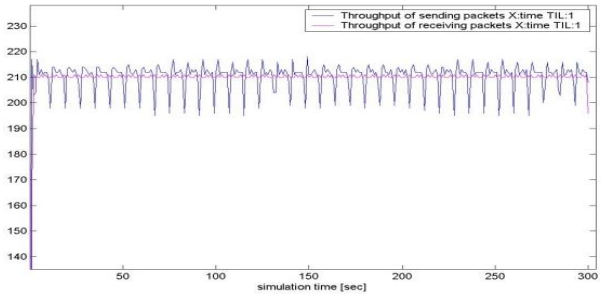

Figure 13. Packet starts dropping at reduced link size 0.875 Mbps with latency 2.875 ms for a four flow network

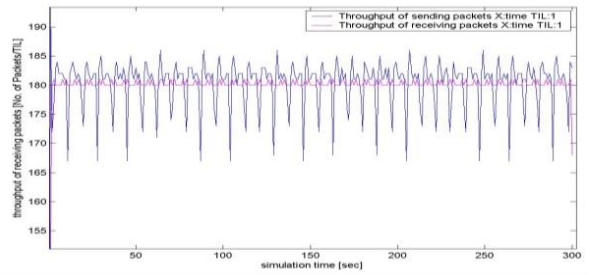

Figure 14. Frequency of packet drop increases at reduced link size 0.750 Mbps with latency 2.875 ms for a four flow network.

The figure below shows the reduced link size and latency of the bottleneck for a three flow network. Calculations and simulation results have shown that using this topology the throughput receiving rate becomes 99.78%. The simulation result also suggests that the improvement achieved over algorithm from [19] has increased by 0.37%. As the SDLM algorithm reaches the limit step size only an upper and middle bound is found.

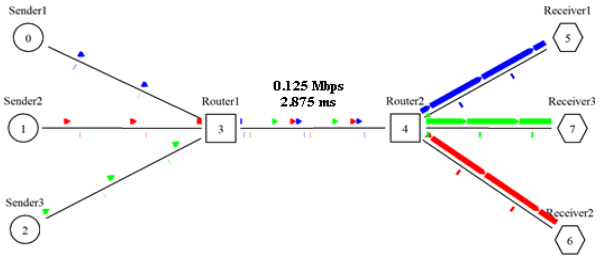

Figure 15. Reduced bottleneck link size and delay for three flow fixed network.

Figure 16 shows that at link size 0.250 Mbps with latency 2.875 ms the packet does not drop due to the balance of adequate amount of information respective to the bottleneck capacity. Figure 17 shows that at link size 0.125 Mbps and delay of 2.875 ms the packet starts dropping due to the flow of information which exceeds the bottleneck capacity.

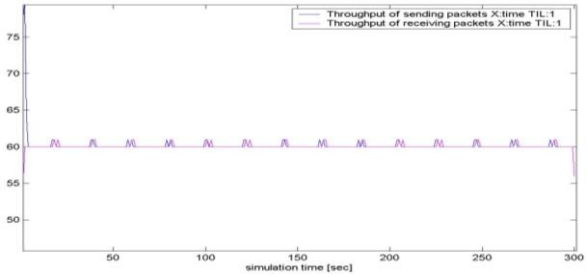

Figure 16. Packet stops dropping at reduced link size 0.250 Mbps with latency 2.875 ms for a three flow network.

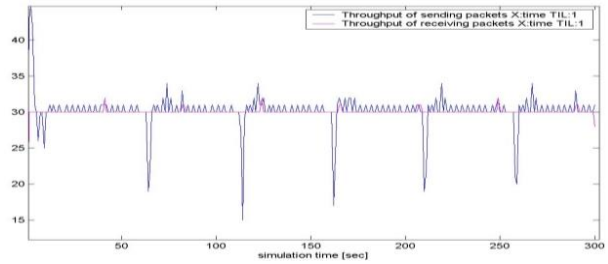

Figure 17. Packet starts dropping at reduced link size 0.125 Mbps with latency 2.875 ms for a three flow network

The figure below shows the reduced link size and latency of the bottleneck for a two flow network. Calculations and simulation results have shown that using this topology the throughput receiving rate becomes 100.00%. The simulation result also suggests that the improvement achieved over algorithm from [19] has increased by 0.29%.

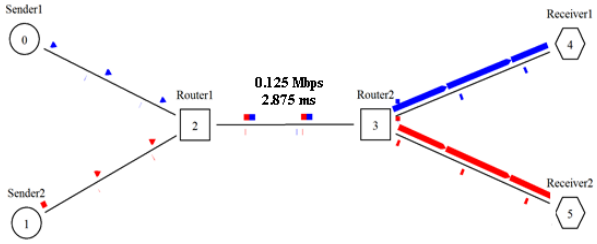

Figure 18. Reduced bottleneck link size and delay for two flow fixed network

Figure 19 shows the experiments with 2 flows with different link size and latency in the single bottleneck link respectively. The figure shows that at link size 0.125 Mbps with latency 2.875 ms the packet does not drop due to the balance of adequate amount of information respective to the bottleneck capacity. The lower bound could not be retrieved as the algorithm reaches the limit step size.

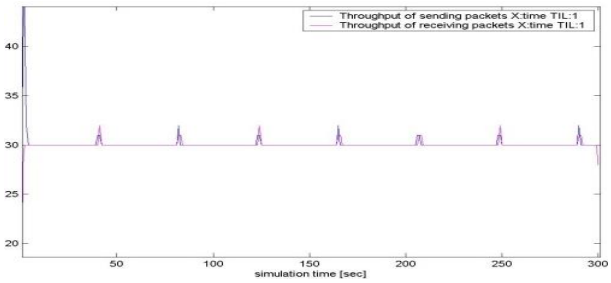

Figure 19. Packet stops dropping at reduced link size 0.125 Mbps with latency 2.875 ms for a two flow network.

Table 2 shows the Total throughput sent, received and dropped in the same bottleneck link for 2,3,4 and 5 flows. The same datum has been represented by packets and KiloBytes (KB). Table 3 gives an overall comparison between the AIMD algorithm presented in [19] and RLO-SDLM algorithm. It is observed that an average improvement of 0.52% has been achieved by implementing RLO-SDLM algorithm on the same topology mentioned in [19].

TABLE 2. Total throughput representation of information sent, received and dropped in packed and KB. According to the NS2 simulation 1 packet is approximately 540.2002 bytes

# of flows	Total Through put sent (packets)	Total Through put sent (KB)	TCP Total Through put sent received (packets)	Total Through put sent received (KB)	Total Through put drop (packet)	No. of packets drop (KB)
2	9043	4773.879	9043	4773.879	0	0
3	9078	4799.643	9058	4789.069	20	10.574
4	63146	33319.684	63108	3329 9.633	38	20.051
5	135189	71377.505	135131	7128.907	58	30.597

TABLE 3. The comparison between RLO-SDLM algorithm and AIMD in [19]

# of flows	Rate of throughput received RLO-SDLM	Rate of throughput received [19]	Improvement achieved	Average Achievement (Overall)
2	100%	99.71%	0.29%	
3	99.78%	99.41%	0.37%	
4	99.94%	99.73%	0.21%	0.52%
5	99.96%	99.37%	0.59%	

TABLE 4. Lower and Upper limit of bottleneck link size and delay for 5 sender network compared to [19]

5 Nodes	Packet Drop	No Packet Drop	No Packet Drop [19]
Link Size (Mbps)	1.875	2.000	5.000
Delay (ms)	2.875	2.875	30.000

TABLE 5. Lower and Upper limit of bottleneck link size and delay for 4 sender network compared to [19]

4 Nodes	Packet Drop	No Packet Drop	No Packet Drop [19]
Link Size (Mbps)	0.875	1.000	5.000
Delay (ms)	2.875	2.875	30.000

TABLE 6. Lower and Upper limit of bottleneck link size and delay for 3 sender network compared to [19]

3 Nodes	Packet Drop	No Packet Drop	No Packet Drop [19]
Link Size (Mbps)	0.125	0.250	5.000
Delay (ms)	2.875	2.875	30.000

TABLE 7. Lower and Upper limit of bottleneck link size and delay for 2 sender network compared to [19]

2 Nodes	Packet Drop	No Packet Drop	No Packet Drop [19]
Link Size (Mbps)	Unknown	0.125	5.000
Delay (ms)	2.875	2.875	30.000

CONCLUSION

In this paper we have presented and evaluated a novel MVL operator RLO and SDLM algorithm for congestion control and management. The hybrid combination of MVL and congestion control algorithm produces RLO-SDLM algorithm which helps decide the upper and lower limit of the link size and latency of the bottleneck. The results obtained using the RLO-SDLM algorithm is compared using a benchmark network consisting of 5 senders and receivers. The results achieved have shown that the RLO-SDLM algorithm outperforms the conventional New AIMD algorithm from [19] and an overall improvement of 0.52% has been achieved using reduced link size and less latency period.

REFERENCES

1. A.K Chowdhury, N. Raj and A.K. Singh, *A Novel High Deduction Algorithm for synthesizing Multiple-Valued Logic and Circuit Realization*, pp. 7-20, (2013) (in review).
2. J. Wang, J. Wen, J. Zhang and Y. Han, *GLOBECOM Workshops (GC Wkshps)*, pp.2065-2069, IEEE (2010).
3. C. Lai, K.C. Leung and V.O.K Li, *Design and Analysis of TCP AIMD in Wireless Networks*,Wireless Communications andNetworking Conference (WCNC), IEEE, Shanghai, 2013, pp.1422,1427.
4. K.C. Leung and V.O.-K. Li, *IEEE Communications Surveys & Tutorials*, **8**, 64-79 (2006).
5. S. Xiaoling, *8th International Conference of Information Science and Digital Content Technology (ICIDT)*, pp.703-706, Jeju, Korea (2012).
6. D.F Kassa, and S. Wittevrongel, *25th IEEE International Performance, Computing, and Communications Conference. IPCCC*, pp.9 and pp.366, Arizona, U.S.A (2006).
7. J. Wang, J. Wen, J. Zhang and Y. Han, *Proceedings 2011 IEEE INFOCOM*, Shanghai, China, pp.2894,2902 (2011).
8. Y.Li, R. Zhang, Z. Liu,and O. Liu, *2nd IEEE International Conference on Broadband Network & Multimedia Technology IC-BNMT*, pp.786-790, Beijing, China (2009).
9. M. Zhou, Fifth International Conference Computational and Information Sciences (ICCIS), pp.1130,1133, Shiyan, Hubei Province, China (2013).
10. P. Lablan, *Multi-Valued Logic*, http://archive.is/rIyG (2005).
11. K.C Smith, (1988), *Computer*, **21**, pp.17-27 (1988).
12. M. Abd-El-Barr, L. Al-Awami, *Proceedings of the 15th International Conference on Microelectronics*. pp. 308-312 (2003).
13. H. Nakahara, T. Sasao and M. Matsuura, *41st IEEE International Symposium on Multiple-Valued Logic*. pp. 125-130, Tuusula, Finland (2011).
14. A.K, Jain, R.J Bolton and M. Abd-El-Barr, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, **40**, pp. 515-522 (1993).
15. A.K, Jain, R.J Bolton and M. Abd-El-Barr, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, **40**, pp. 503-514(1993)
16. A.D. Gawande and S.A. Ladhake, *7th WSEAS International Conference on Instrumentation, Measurement, Circuits and Systems*, pp. 108-113, Hnagzhou, China (2008).
17. T. Temel,A. Morgul and N. Aydin, IEE Proceedings Circuits, Devices and Systems, **153**, pp.489-496 (2006).
18. T. Temel and A. Morgul, *IEEE International Symposium on Circuits and Systems*, **1**, pp. I-881-I-884 (2002).
19. H. J. Natiq, Z. Ahamd, Z. Othman and M. Suhbramaniam "The New AIMD Congestion Algorithm