MCD: An End-to-end Multicast Congestion Detection Scheme using Support Vector Machines

Xiaoming Liu[†], Christian W. Omlin[‡]
[†]Department of Computer Science, University of the Western Cape, Bellville, SOUTH AFRICA
[‡]Middle East Technical University, Northern Cyprus Campus, Güzelyurt, TURKEY
email: *2478875@uwc.ac.za, Christian.Omlin@yahoo.com*

***Abstract*—Congestion is one of the most important issues impeding the development and deployment of IP multicast and multicast application. In this paper, we propose MCD, an end-to-end multicast congestion detection scheme with support vector machines. We focus on using support vector machines to detect incipient congestion in an end-to-end multicast network after training by using structural information about the multicast network. In this way, by using a learning system, we can predict incipient congestion in advance instead of waiting for packet loss. Simulation results show that support vector machine is an appropriate mechanism for decision making in proactive multicast congestion detection.**

***Index Terms*—Multicast, congestion detection, SVM.**

I. Introduction

IP multicast is an efficient mechanism for simultaneously transmitting bulk data to multiple receivers. Many applications can benefit from multicast, such as audio and videoconferencing, multiplayer games, multimedia broadcasting, distance education, and data replication. In this paper, we propose MCD, an end-to-end multicast congestion detection scheme with support vector machines. Congestion is one of the most important issues impeding the development and deployment of IP multicast and multicast applications. Many congestion control schemes have been proposed to tackle the multicast congestion problem. Almost all existing multicast congestion control schemes use packet losses as the indication of congestion. However, few of the schemes focus on using machine learning to predict multicast congestion in advance. Machine learning has already been successfully applied in a number of areas without much background information, and has given useful results. Because we tackle the multicast congestion problem with the end-to-end assumptions, we cannot obtain the opportune and accurate congestion information directly from inside the network. Therefore, machine learning is particularly appropriate due to the absence of congestion information and the unpredictable variance of network congestion. In the case of learning to detect multicast congestion, the output is a simple yes/no tag, i.e. as a binary output value. Thus, multicast congestion detection can be viewed as a binary classification problem. Support Vector Machines (SVMs) are used to detect incipient congestion with great accuracy in an end-to-end multicast network after training by using structural information about the multicast network.

Our proposed scheme, MCD, detects multicast network congestion before packet loss occurs using support vector machines. The scheme is purely receiver-based in the sense that it operates on a stream of multicast data packets from the source without any other support from network elements, source or in the packet format of underlying multicast transport protocols. One earlier work by Li et al [14] uses simple thresholding techniques and "accumulation concept" to detect multicast network congestion without necessarily inducing packet loss. However, the accumulation measurement algorithm is based on aggregated flow information, whereas our work uses detailed flow information for congestion detection. Another single-rate multicast congestion control work by DeLucia et al [10] uses the congestion detection method proposed in TCP Vegas [5], a unicast congestion avoidance scheme. DeLucia et al's scheme only detects incipient congestion on the source side for the paths between the source and representative receivers, when other receivers still detect congestion by packet losses. For comparison, our scheme detects incipient congestion on the receiver side for all paths. In other "congestion control" schemes including PGMCC [16], TFMCC [23] and references within, congestion detection can be implemented by monitoring the number of dropped packets.

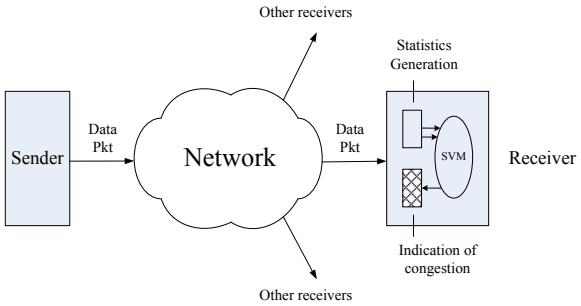

Fig. 1. SVM-based multicast congestion detection model.

In our scheme, SVMs [9] detect incipient congestion on the receiver side (Figure 1). We gather statistics from the stream of multicast data packets as the input data of SVM. The outputs of SVM are collected as congestion indications. Before using SVM classifier to detect multicast congestion, the classifier is trained off-line. For training SVM, labeled data with two labels "congested" and "uncongested" are generated by accumulation measurement algorithm and regression techniques.

Simulation results show that MCD can achieve great accuracy in predicting incipient congestion. Since our congestion detection model is incompatible with that of TCP, we verify the performance of MCD by comparing with an accumulation measurement algorithm [14].

II. Algorithm Description

In MCD, four algorithms are implemented on the receiver side, including an accumulation measurement algorithm, a regression estimation algorithm, a statistics generation algorithm and an SVM classification algorithm. The training dataset is generated by the accumulation measurement algorithm [14] and the regression estimation algorithm. We gather statistics from the stream of multicast data packets for training dataset generation and SVM classification using statistics generation algorithm. After off-line SVM training, incipient congestion is detected using SVM classification. We present the details in the following.

A. Accumulation Measurement

An accumulation measurement algorithm is performed to obtain the training samples used to train the SVM for incipient congestion.

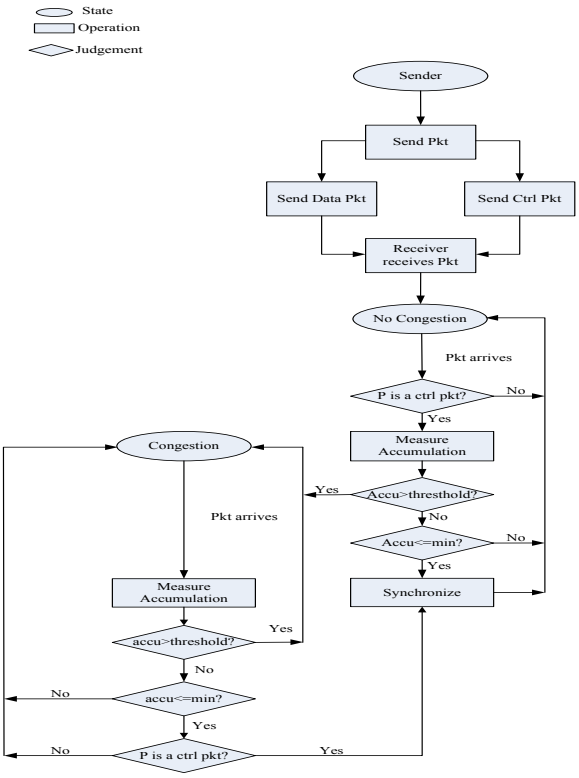

Fig. 2. Accumulation measurement[14].

We borrow the ideas from the MCA scheme [14]. In Li et al's MCA scheme, the concept of accumulation based on unicast is extended to multicast. The accumulation measurement algorithm is explained by Figure 2 and the following specifications.

Similar to Li et al's MCA scheme, the control packets are multicast to all receivers at a fixed time interval. A normal data packet can be easily changed to a control packet by turning on some one-bit flag and adding its sending time into the optional field. On the receiver side, accumulation is measured relying on the control packets. If accumulation is larger than two packets, congestion occurs, and then the time is recorded. After that, we use the regression technique to obtain the estimation value of the statistics about the multicast stream when incipient congestion is detected.

B. Regression Estimation

In the period of training set generation, we gather two types of labeled data ("congested" and "uncongested") as training sample for SVM classification. On the receiver side, accumulation is measured to detect congestion as every control packet arrives. Using the regression technique, we can calculate the estimation values of the statistic about the multicast stream according to the control packet arrival time.

Although the control packet arrival time can be collected by measuring accumulation, it is difficult to measure directly the statistics about the multicast stream at the moment. Thus we use the regression estimation algorithm to obtain the estimation values of the statistic. During training set generation, the number of packets received at fixed intervals (40ms) is recorded. Two variables can be obtained from the sampling, namely the number of packets received at fixed intervals $Y_i = \{Y_1, Y_2, \ldots, Y_n\}$ and the time of sampling $X_i = \{X_1, X_2, \ldots, X_n\}$ where n is sampling number before the control packet arrives. We assume that the regression line of variable Y, on variable X has the form $\beta_0 + \beta_1 X$. Then we can write the linear regression model

$$Y = \beta_0 + \beta_1 X + \varepsilon, \tag{1}$$

where β_0 and β_1 are the parameters of the model, and ε is the increment by which any individual Y may fall off the regression line.

The slope of the fitted straight line [12] is thus

$$\begin{aligned} b_1 &= \frac{\sum X_i Y_i - [(\sum X_i)(\sum Y_i)]/n}{\sum X_i^2 - (\sum X_i)^2/n} \\ &= \frac{\sum (X_i - \overline{X})(Y_i - \overline{Y})}{\sum (X_i - \overline{X})^2} \end{aligned} \tag{2}$$

where all summation are from $i = 1$ to n. The fitted regression equation [12] is

$$\hat{Y} = \overline{Y} + b_1 (X - \overline{X}) \tag{3}$$

where $\hat{Y}$ is the estimation value of Y (the number of packets received at a fixed interval as the control packet arrives); X is the time of sampling in the period of statistics generation; b_1 is given by Equation (2). And we have

$$\begin{aligned} \overline{X} &= \frac{1}{n} \sum_{i=1}^{n} X_i, \\ \overline{Y} &= \frac{1}{n} \sum_{i=1}^{n} Y_i. \end{aligned} \tag{4}$$

According to Equations (2, 3, 4, 5), we can calculate the estimated number of packets received at a fixed interval ($\hat{Y}$) when the control arrives.

C. Statistics Generation

In our scheme, the statistics are collected on the receiver side for constructing the training set as well as the working set. We can obtain the statistics – the sample mean and variance – using simple statistical techniques. The details of this process are presented in the following discussion.

1) Statistics Generation for Training Set: The training set is generated by using two types of training sample labeled as "congested" or "uncongested". As shown in Figure 3, incipient congestion is detected by measuring accumulation when the control packet arrives; the estimated number of packets, relating to the control packet arrival time, is then computed using the regression estimation algorithm. Depending on the estimated number of packets, the estimated statistics can be obtained for training set generation when every control packet is received. The estimated statistics are labeled as "congested" or "uncongested" according to the detection results from the accumulation measurement.

On the receiver side, the number of packets received at fixed intervals (40ms) is recorded as sample during training set generation. Two parameters $\overline{Y_k}$ and S_k are computed as the estimated statistics when every control packet is received. We have,

$$\overline{Y_k} = \frac{1}{n+1} \sum_{i=1}^{n+1} Y_i + \frac{\hat{Y}}{n+1} \tag{5}$$

where $\overline{Y_k}$ is the estimated mean of sample (the number of packets received at fixed intervals); Y_i is the number of packets received at fixed intervals before the control packet arrives; n is sampling number before the control packet arrives. According to the unbiased formula

$$S = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (X_i - \overline{X})^2}, \tag{6}$$

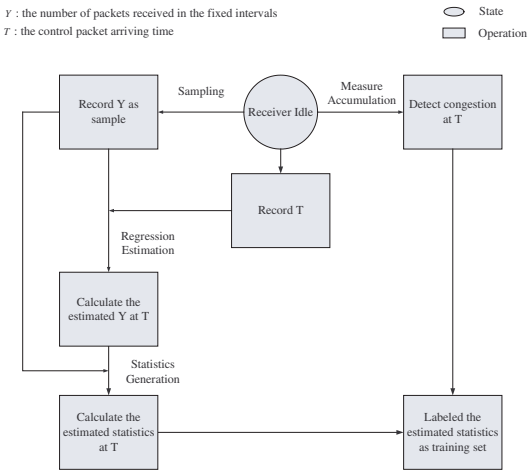

Fig. 3. Training set generation.

we have,

$$S_k = \sqrt{\frac{1}{n}\sum_{i=1}^{n+1}(\hat{Y} - \overline{Y_k})^2} \tag{7}$$

where S_k is the estimated standard deviation (variance); $\hat{Y}$ is the estimation value as the control packet arrives; $\overline{Y_k}$is the estimated mean of sample (the number of packets received at fixed intervals); n is sampling number before the control packet arrives. The estimated mean of sample $\overline{Y_k}$ and the estimated standard deviation (variance) S_k will be computed to create the training dataset where our SVM learns about incipient congestion.

2) Statistics Generation for Working Set: During SVM classification, the number of packets received at fixed intervals (40ms) is recorded as a sample at the receiver side. Two parameters $\overline{Y_t}$ and S_t are computed as the statistics of the multicast stream when every sample is recorded. The unlabeled data is also the working set of SVM classification. We have,

$$\overline{Y_t} = \frac{1}{n}\sum_{i=1}^{n} Y_t \tag{8}$$

where $\overline{Y_t}$ is the sample mean (the number of packets received at fixed intervals); Y_t is the number of packets received at fixed intervals; n is sampling number.

The unbiased formula is also used for sample variance. We have,

$$S_t = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(Y_t - \overline{Y_t})^2} \tag{9}$$

where S_t is the sample standard deviation (variance); $\overline{Y_t}$ is the sample mean (the number of packets received at fixed intervals), Y_t is the number of packets received at the fixed intervals and n is sampling number.

D. Support Vector Machines

Support Vector Machines [9] is one type of learning method constructed by Vapnik and Cortes. It is used in our scheme to detect incipient congestion that occurs within a time series. We chose to use the SVM since it has been successfully applied in many fields, such as word sense disambiguation, text classification, part-of-speech tagging, web page classification, and question classification [22]. SVM first maps input space into some high dimensional feature space, then constructs a linear decision surface in this feature space that relates to a non-linear decision surface in the original input space. The comparison between SVM and other classification

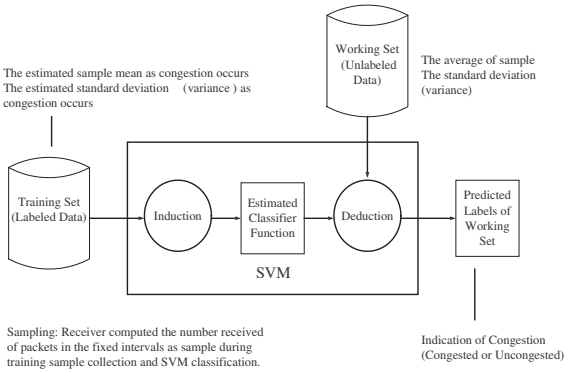

Fig. 4. SVM Learning.

methods is given in [19], [18], [2]. In our scheme, Vapnik's C-Support Vector Classification algorithm [22] is used to classify the statistics of multicast stream and arbitrate the congestion according to the training sample. We summarize the algorithm here.

Our training data consists of N pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, with $x_i \in R^2$ and $y \in \{-1, 1\}$. The primal form considered is

$$\begin{aligned} \min_{w,b,\xi} \ & \frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \ & y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \qquad i = 1, \ldots, l. \end{aligned} \tag{10}$$

It can be rephrased as

$$\begin{aligned} \min_{\alpha} \ & \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ & 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, l, \\ \text{subject to} \ & y^T\alpha = 0, \end{aligned} \tag{11}$$

where e is the vector of all ones, $C > 0$ is the upper bound, Q is an $l \times l$ positive semi-definite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, and $K(x_i, x_j) \equiv \phi(x_i)^T\phi(x_j)$ is the kernel. Here training samples x_i are mapped into a higher dimensional space by the function ϕ. The decision function is

$$sgn(\sum_{i=1}^{l} y_i\alpha_i K(x_i, x) + b). \tag{12}$$

III. Scheme Description

As it is depicted in Figure 4, the receivers collect the training samples for estimating the classifier function using accumulation measurement and regression estimation algorithm. After off-line SVM training, we calculate the statistics of the multicast stream at fixed intervals as a working set (unlabeled data) for SVM classification, and finally by using the estimated function, we classify the unlabeled data (working set) when each time a new statistic is generated. According to the result of SVM classification, we can detect incipient congestion on the receiver side. Therefore, the MCD scheme can be split into two operating parts, viz. training set generation and SVM Classification. We present the details in the following.

A. Training Set Generation

To obtain the training samples where our SVM learns about incipient congestion, accumulation measurement and regression computation are executed on the receiver side. The sender multicasts data and control packets to the receivers, and the receivers obtain the congestion time before packet loss occurs using an accumulation measurement algorithm. Meanwhile, the number of packets received

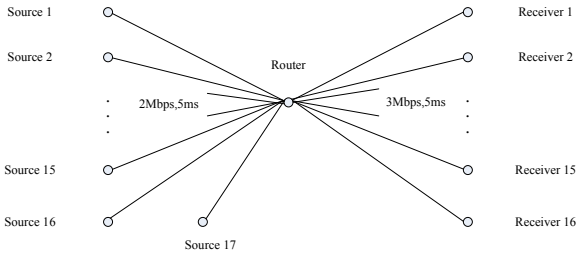

Fig. 5. 16-Receiver Star Topology.

at fixed intervals (40ms), it is same with the transmission interval of control packets) is recorded as the sample on the receiver side. The estimation value of the number of packets received is calculated using the regression algorithm while every control packet arrives. And then, we can compute the sample mean and variance depending on the estimated number of packets. After that, the estimated statistics are labeled as "congested" or "uncongested" based on the detection result of accumulation measurement. Our training set can be generated using the labeled statistics.

B. SVM Classification

The sender multicasts data packets to the receivers after our SVM is trained. On the receiver side, the number of packets received at fixed intervals (40ms) is sampled. Two parameters, the average of number of packets received at fixed intervals and variance, are computed as the statistics of the multicast stream when every sample is recorded. The unlabeled data is also the working set of SVM classification. The C-Support Vector Classification algorithm [22] is used to classify the working set and arbitrate the congestion.

IV. Simulations and Experiments

To verify the performance of our scheme, we ran several `ns-2` [1] simulations and statistical experiments. We collect samples and record every control packet arrival time on `ns-2`, and generate the training set by statistical experiments. LIBSVM is used to train and optimize our SVM. After off-line training, we use the SVM classifier to detect incipient multicast congestion.

A. Sample Collection

In all `ns-2` simulations, the data packet size is 1,000 bytes, the bottleneck buffer size is 10K bytes, the initial RTT (round trip time) is 100 milliseconds. The simulation time is 60 seconds. The traffic generator is Pareto. We first collect samples and record every control packet arrival time on the simple topology in Figure 5. We used the star topology to generate asynchronous and independent congestion on different paths. There are 16 end nodes in the topology. Between each pair of source i and receiver i $(i = 1 \cdots 16)$, there are three TCP Reno flows. Furthermore, there is a multi-receiver MCD flow from source 17 to all 16 receivers. Therefore, on a path between the router and any receiver, the multi-receiver MCD flow competes with three TCP flows. On the receiver side, the number of packets received at fixed intervals – 40ms, it is same with the transmission interval of control packets – is recorded as the sample; every receiver obtains 1,476 samples. 16,925 control packets are multicasted to receivers, and every control packet arrival time is gathered for training set generation. The time samples are labeled as "congested" or "uncongested" based on the detection result of accumulation measurement. And their statistics are as shown in Table 1.

B. Training Set Generation

After obtaining the samples from `ns-2`, we generate the training set for our SVM using the statistical methods. The estimation value of the number of packets received is calculated using the regression algorithm when every control packet arrives. We then compute the sample mean and variance depending on the estimated number of packets. The estimated statistics are labeled as "congested" or "uncongested" depending on the labeled time sample (control packet arrival time). Our training set can be generated using the labeled statistics. There are 16,925 records in the training set which fall into two classes. Each record has two attributes, viz. the estimated sample mean and variance.

TABLE I. Statistics of the Time Sample during Sample Collection.

Receiver No.	Control Pkts Received	Uncongested	Congested
1	1,066	811	255
2	1,025	788	237
3	1,043	798	245
4	1,084	821	263
5	1,046	811	235
6	1,049	810	239
7	1,070	826	244
8	1,057	820	237
9	1,032	796	236
10	1,086	838	248
11	1,060	822	238
12	1,074	815	259
13	1,070	822	248
14	1,053	812	241
15	1,081	840	241
16	1,029	785	244
Sum	16,925	13,015	3,910

C. SVM Training

Chang et al's LIBSVM [8] is used to train and optimize our SVM. LIBSVM is an integrated software for support vector classification (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). To choose the best parameters of our SVM, `grid.py` [25], a model selection tool, is used for C-SVM classification. It uses cross validation (CV) technique to estimate the accuracy of each parameter combination in the specified range. During training period, parameter C was set to 32,768 and parameter γ was set to 8. After training is finished, the cross validation accuracy is 90.9%.

D. Detect Congestion using SVM Classification

After off-line training, an `ns-2` simulation is run to test the performance of MCD. LIBSVM is used for SVM training and classification. The same `ns-2` configuration used by sample collection is implemented during SVM classification. The simulation time is again 60s. But the sender only multicasts data packets to receivers, and then the receivers detect incipient congestion using SVM classification instead of accumulation measurement. Every receiver collects 1,466 samples during the simulation. The sampling time is labeled as "congested" or "uncongested" depending on the classification result of SVM. And their statistics are as shown in Table 2. The congestion time gathered by measuring accumulation is compared with collecting by MCD, as shown in Table 3.

We compare the detection results (the number of congestions and the time of congestion) from MCD with the results collected by measuring accumulation. Figure 6 clearly demonstrates that MCD can achieve great accuracy in predicting incipient congestion by SVM classification. However, at the Receiver 3 and Receiver 5, SVM detects incipient congestions before using accumulation measurement. The data packets will not be lost because we assume that there is another congestion avoidance module which should be able to adjust transmission rate while incipient congestion occurs. But at the Receiver 14, it will possibly suffer congestion while SVM detects incipient congestions with delay. The accuracy of our SVM classification is about 90%. Failing includes either missing congestion or predicting congestion while there is none.

TABLE II. Statistics of MCD Sampling

Receiver No.	Sampling Number	Uncongestion	Congestion
1	1,466	1,115	351
2	1,466	1,144	322
3	1,466	1,217	249
4	1,466	1,135	331
5	1,466	1,175	291
6	1,466	1,142	324
7	1,466	1,170	296
8	1,466	1,170	296
9	1,466	1,121	345
10	1,466	1,162	304
11	1,466	1,154	312
12	1,466	1,138	328
13	1,466	1,157	309
14	1,466	1,064	402
15	1,466	1,134	332
16	1,466	1,136	330
Sum	23,456	18,334	5,122

TABLE III. Number of Congestions Compared with MCD and Accumulation Measurement

Receiver No.	MCD	Accumulation Measurement
1	351	255
2	322	237
3	249	245
4	331	263
5	291	235
6	324	239
7	296	244
8	296	237
9	345	236
10	304	248
11	312	238
12	328	259
13	309	248
14	402	241
15	332	241
16	330	244
Sum	5,122	3,910

Chang et al's LIBSVM is used to train and optimize our SVM. LIBSVM is an integrated software for support vector classification (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). We use a model selection tool, grid.py, for our C-SVM classification. During training period, parameter C was set to 32,768 and parameter γ was set to 8. After training is finished, the cross validation accuracy is 90.9%.

We changed the seed for the random number generator (RNG) of the traffic generator in the `ns-2` simulations. The experiment was repeated nine times. We estimated the effect of MCD scheme by comparing between two congestion curves. Their statistics and effects are as shown separately in Table 4 and Table 5. Some receivers will possibly suffer congestion while SVM detects incipient congestions with a clear delay. The percentage of the receivers which can detect incipient congestions in time by using SVM is calculated for each experiment. We then calculate its confidence interval. We also calculate the cross validation accuracy for each experiment.

V. Conclusion

We have proposed MCD, an end-to-end single-rate multicast congestion detection scheme. Using SVM classification, MCD can detect network congestion before packet loss occurs on the receiver side. In this work, we have illustrated that a SVM can achieve great accuracy in predicting congestion. MCD estimates the congestion of network before packet loss occurs at each receiver using SVM. Therefore, MCD does not require ACK-equivalent and NAK-equivalent feedback from receivers. And MCD has no computa-

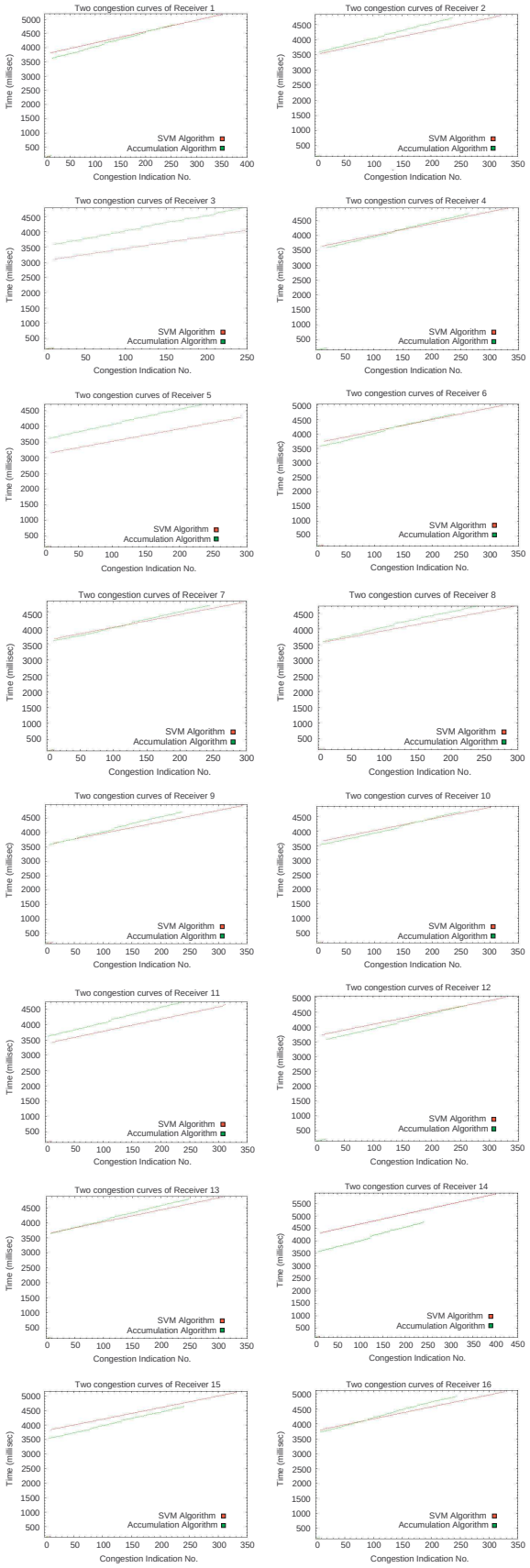

Fig. 6. Two Congestion Curves from MCD and Accumulation Measurement at Receiver 1-16 (During the simulations, the congestion time is recorded with a serial number when each time an incipient congestion indication is generated. The congestion curve can be generated from the dataset with two variables, congestion time and congestion indication serial number.)

Seed of the RNG	The number of receivers which donot suffer incipient congestions	The number of receivers which possibly suffer incipient congestions	The percentage of the receivers which donot sufferincipient congestions	SVM Classification Parameters		Cross Validation Accuracy
			%	c	g	%
2	15	1	93.3333	32768	8.0	90.8983
4	16	0	100.0000	32768	8.0	85.9294
5	16	0	100.0000	32768	8.0	92.1614
8	14	2	85.7143	32768	8.0	89.9976
10	16	0	100.0000	32768	8.0	90.0565
314	15	1	93.3333	32768	8.0	87.4199
456	16	0	100.0000	32768	8.0	88.2545
687	12	4	66.6667	32768	8.0	86.9227
894	14	2	85.7143	32768	8.0	83.9995
910	12	4	66.6667	32768	8.0	82.7727
Average			89.1429			87.8413
Stdev			13.0831			3.0343
Size			10.0000			10.0000
Alpha			0.0500			0.0500
Confidence			8.1088			1.8806

TABLE IV. Statistics of ten MCD simulation experiments with different seeds of RNG.

Seed of the RNG	The effect of MCD scheme Mark O: The receiver does not suffer incipient congestion. Mark X: The receiver possibly sufferes incipient congestion.															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	O	O	O	O	O	O	O	O	O	O	O	O	O	X	O	O
4	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
5	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
8	O	O	O	O	O	O	O	O	X	O	O	O	X	O	O	O
10	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
314	O	O	O	O	O	X	O	O	O	O	O	O	O	O	O	O
456	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
687	O	X	O	X	O	O	O	O	O	O	O	O	O	X	X	O
894	O	O	O	O	O	O	O	O	O	O	O	X	O	O	X	O
910	O	O	O	O	X	X	O	X	O	O	O	O	O	X	O	O

TABLE V. Effect of ten MCD simulation experiments with different seeds of RNG.

tion procedure for congestion estimation at the source. While our scheme detects the congestion without introducing packet loss, other congestion control approaches only do the simplest packet loss detection at receivers. Some approaches with machine learning have been applied to tackle network congestion management problems, such as Thottan's work [21], [20] and Bivens' scheme [4]. In comparison, our scheme is a receiver-based end-to-end multicast congestion detection scheme. It does not require special support from other network management protocols, such as SNMP [6], [7]. Furthermore, MCD uses the average number of packets and variance as the input data for SVM classification, whereas Bivens' scheme [4] collects the traffic patterns (packet delays and the higher moments of the delay distribution) for a collection of directly interconnected routers. They also use a neural network for their more complex input, while SVM is implemented for the proactive multicast congestion detection in our scheme. We realize that many problems exist for which this approach is applicable, predicting congestion is the first step towards avoiding congestion.

Network congestion is a problem that changes very quickly. Any algorithm detecting congestion would have to render the decision before the problem has changed to a degree that the decision is no longer relevant to the environment. Once trained, SVM can render decisions very quickly. Therefore, SVM is an appropriate mechanism for decision making in proactive multicast congestion detection, and should be the target of more research. However, we would like to do more testing with other machine learning algorithms in the future, such as neural networks, decision tree, etc.

REFERENCES

[1] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, M. Handley, A. Helmy, J. Heidemann, P. Huang, S. Kumar, S. McCanne, R. Rejaie, P. Sharma, K. Varadhan, Y. Xu, H. Yu, and D. Zappala, "Improving Simulation for Network Research", *Technical Report 99-702b*, University of Southern California, March 1999, revised September 1999.

[2] K. P. Bennett, and J. A. Blue, "A Support Vector Machine Approach to Decision Tree," *R.P.I. Math Report*, No. 97-100, Rensselaer Polytechnic Institute, Troy, NY, 1997.

[3] J. A. Bivens, "Distributed Framework for Deploying Machine Learning in Network Management and Security," *PhD thesis*, Computer Science, Rensselaer Polytechnic Institute, February 2003.

[4] J. A. Bivens, B. K. Szymanski, and M. J. Emerechts, "Network Congestion Arbitration and Source Problem Prediction using Neural Networks," *Smart Engineering System Design*, 4: 243-252, 2002.

[5] L. Brakmo, and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE JSAC*, 13(8): 1465-1480, October 1995.

[6] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)," *RFC 1905*, January 1996.

[7] J. Case, R. Mundy, D. Partain, and B. Stewart, "Introduction to Version 3 of the Internet-standard Network Management Framework," *RFC 2570*, May 1999.

[8] C. C. Chang, and C. J. Lin, "LIBSVM: A Library for Support Vector Machines," *Technical Report*, Computer Science and Information Engineering, National Taiwan University, 2001-2004.

[9] C. Cortes, and V. N. Vapnik, "Support Vector Networks," *Machine Learning*, 20: 273-297, 1995.

[10] D. DeLucia, and K. Obraczka, "A Multicast Congestion Control Mechanism Using Representatives," *Proceedings of the IEEE ISCC* 1988.

[11] A. Demiriz, "Learning with Capacity Control: A Semi-supervised Approach," *PhD thesis*, Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, June 2000.

[12] N. Draper, and H. Smith, "Applied Regression Analysis, Second Edition," *John Wiley & Sons, Inc.*, ISBN 0-471-029995-5, 1996.

[13] J. Li, "End-to-end Multicast Congestion Control and Avoidance," *PhD thesis*, Computer Science, Rensselaer Polytechnic Institute, July 2003.

[14] J. Li, and S. Kalyanaraman, "MCA: An End-to-end Multicast Congestion Avoidance Scheme with Feedback Suppression," *Computer Communications*, 27(13): 1264-1277, 2004.

[15] D. S. Moore, and G. P. McCabe, "Introduction to the Practice of Statistics, Third Edition," *W. H. Freeman and Company*, ISBN 0-7167-3502-4, 1998.

[16] L. Rizzo, "PGMCC: A TCP-friendly Single-Rate Multicast Congestion Control Scheme," *SIGCOMM 2000*, August 2000.

[17] V. K. Rohatgi, "Statistical Inference," *John Wiley & Sons, Inc.*, ISBN 0-471-87126-5, 1984.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel distributed processing: Explorations in the macrostructures of cognition*, Cambridge, MA, Bradford Books, 1: 318-362, 1986.

[19] B. Schölkopf, K. K. Sung, C. J. C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparing Support Vector Machines with Gaussian Kernel to Radial Basis Function Classifiers," *IEEE Transactions on Signal Processing*, 45(11): 2758-2765, 1997.

[20] M. Thottan, and C. Ji, "Adaptive Thresholding for Proactive Network Problem Detection," *IEEE International Workshop on Systems Management*, Pages 108-116, Newport, Rhode Island, April 1998.

[21] M. Thottan, and C. Ji, "Proactive Anomaly Detection using Distributed Intelligent Agents," *IEEE Network, Special Issue on Network Management*, 12(5):21-27, September-October 1998.

[22] V. N. Vapnik, "Statistical Learning Theory," *Wiley InterScience*, 1998.

[23] J. Widmer, and M. Handley, "Extending Equation-based Congestion Control to Multicast Applications," *SIGCOMM 2001*, August 2001.

[24] http://people.revoledu.com/kardi/tutorial/Statistics/Variation.htm

[25] http://www.cs.rpi.edu/~lij6/Research/ormcc/ormcc.html

[26] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[27] http://www.multicasttech.com/status/

Xiaoming Liu received the M.Sc. degree in Computer Science from the University of the Western Cape (UWC), South Africa, in 2008. He is currently working toward the Ph.D. degree in Computer Science at the University of the Western Cape (UWC), South Africa. His research interests include intelligent wireless network management with situation awareness and AI methods, distributed computing, and time series modeling.

Christian Omlin Christian Omlin is Visiting Professor of Computer Engineering at the Middle East Technical University (Northern Cyprus Campus). His research interests include the application of machine learning methods to intelligent information retrieval and service delivery on the World Wide Web, knowledge-based neurocomputing and hybrid systems, integration of verbal an signed communication and biometrics.