DC-Vegas : A Delay-based Congestion Control Algorithm

Asif Shahriar
#1805040, CSE, BUET
Email: 1805040@ugrad.cse.buet.ac.bd

Acknowledgements

This experiment was conducted as part of the term project of CSE321 - Computer Networking Sessional course in BUET. It was performed under the supervision of Meher Afroz, part-time lecturer, Department of CSE, BUET.

***Abstract*—To achieve the excellent performances of DC-TCP congestion control algorithms in datacenters at the relatively low deployment cost of TCP-Vegas algorithm, we proposed a delay-based congestion control algorithm in this paper; namely DC-Vegas. NS-2 simulation shows that DC-Vegas can produce better throughput and packet delivery ratios than TCP-Vegas while the average end-to-end delay is just slightly higher.**

I. Introduction

Traditional TCP congestion control algorithms designed for the Internet environments have encountered many problems in datacenter networks, like soft latency and huge throughput loss in incast scenarios. To address these issues many improved solutions have been proposed. Although these solutions achieved excellent performance in laboratorial datacenter networks, there still exist some deployment barriers when adopting them in existing datacenters that were already built all over the world. Algorithms such as DCTCP require ECN (Explicit Congestion Notification) support for congestion detection, a capability that is not universally available even today. On the other hand, delay-based TCP Vegas algorithm performs well for reducing transmission latency, but poorly in incast scenarios. Based on this insight, in this paper we propose the DC-Vegas algorithm, which combines the low-cost deployment advantage of TCP Vegas and the excellent performance of DCTCP in datacenters. We have simulated the performance of DC-Vegas using the Network Simulator-2 and compared it against TCP-Vegas. We have run the simulation on a wired network and a wireless network. The performance metrics used to compare these two algorithms are mentioned in brief in section IV. The algorithms are discussed briefly in section V. The simulation result, along with graphs, are presented in section VI. We point out some of the limitations of this simulation experiment in section VII. Finally, the findings are presented in a summarized form in section VIII.

II. Network Topology under Simulation

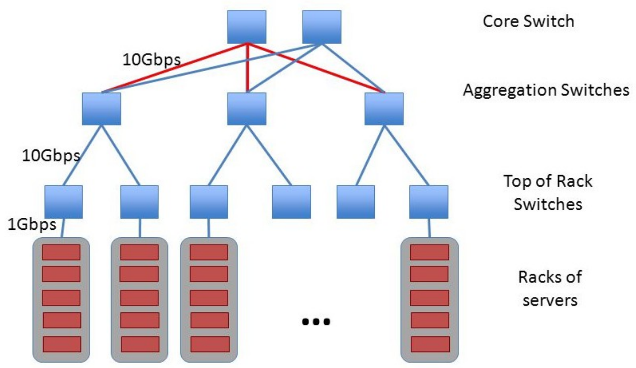


Fig. 1. Simulation topology

III. Parameters under Variation

Parameters	Wired/Wireless	Values
Number of nodes	both	42, 62, 82, 102, 122
Number of flows	both	10, 20, 30, 40, 50
Packets per second	both	Wired: 1.0 mb, 1.25 mb, 1.5 mb, 1,75 mb, 2.0 mb Wireless: 1.0 mb, 2.0 mb, 3.0 mb, 4.0 mb, 5.0 mb
Speed of node	wireless only	5 m/s, 10 m/s, 15 m/s, 20 m/s, 25 m/s

IV. Performance Metrics

Network Throughput: The amount of data that can be successfully transmitted over a network within a specific period of time, typically measured in bits per second (bps) or bytes per second (Bps).

End-to-end Delay: The amount of time it takes for a packet to travel from the source to the destination over a network.

Packet Delivery Ratio: Ratio of the number of packets delivered to end destination to the total number of packets sent.

Packet Drop Ratio: Ratio of the number of packets dropped to the total number of packets sent.

Total Energy: Total amount of energy consumed by all the devices and components that make up the network over the simulation period of time, measured in Joule (for wireless network only).

Average Energy per Packet: The amount of energy consumed by a device to transmit or receive a single packet of data on average, measured in Joule (for wireless network only).

Per Node Throughput: The amount of data that can be transmitted or received by a single node in the network over the simulation period, measured in bps.

Per Node Residual Energy: The amount of energy remaining in the battery of a single node in the network after the simulation ends, measured in Joule.

V. Algorithms

A. *TCP-Vegas Algorithm*

TCP Vegas uses end-to-end queue length samples to detect congestion. In each RTT, TCP Vegas estimates current queue length as:

$$q = cwnd - cwnd \times \frac{RTT_{min}}{RTT} \tag{1}$$

where $cwnd$ is the congestion control window size, RTT is the sampled RTT and RTT_{min} is the minimum observed RTT. TCP Vegas adjusts the window size in each RTT by comparing q with two threshold values, k_a and k_b. If $k_a < q < k_b$, then TCP-Vegas nothing. Otherwise, if $q > k_b$ then TCP-Vegas considers the network congested and reduces the window but instead increases the window if $q < k_a$.

B. *DC-Vegas Algorithm*

The binary congestion detection method of TCP-Vegas does not work well in datacenters which suffers from frequent incast scenarios. In DC-Vegas, when an ACK arrives, the sender first estimates current network queue length q by (1) and then compares q with a threshold k_{dcv}. When all packets in the same window are acknowledged, DC-Vegas calculates

$$F_{dcv} = \frac{\text{number of ACKs with } q > k_{dcv}}{\text{Total number of ACKs in the window}} \tag{2}$$

Then DC-Vegas users the value of F_{dcv} to find

$$\alpha_{dcv} = (1-g) \times \alpha_{dcv} + g \times F_{dcv} \tag{3}$$

where g is a constant with $0 < g < 1$. Here, α_{dcv} is used to measure network congestion level.

DC-Vegas updates its window $cwnd$ in each RTT according to the following equation:

$$cwnd = \begin{cases} cwnd - \frac{\alpha_{dcv}}{2} & \text{if } F_{dcv} > 0 \\ cwnd + \frac{1}{cwnd} & \text{if } F_{dcv} = 0 \end{cases} \tag{4}$$

Thus, when network congestion is at a low level, DC-Vegas increases its window by a small value; otherwise, the window is reduced significantly to alleviate high network congestion.

VI. Simulation Result

A. *For Wired Network*

For the wired network, we followed a typical datacenter topology. There were two core switches, each of which was connected to 10 aggregation switches. Each aggregation switch was connected to 2 top rack switches. Then there were typical servers. To simulate incast scenarios, we fixed a destination device and several senders were sending chunks of packet to that destination.

As we can see from figures 2, 3, 4, the throughput in DC-Vegas is almost identical to TCP-Vegas in the absent of incasts. The end-to-end delay stays roughly the same, too. So DC-Vegas was able to match the good delay performance of TCP-Vegas. In terms of accuracy, DC-Vegas performed better than TCP-Vegas with a higher packet delivery ratio and lower drop ratio.

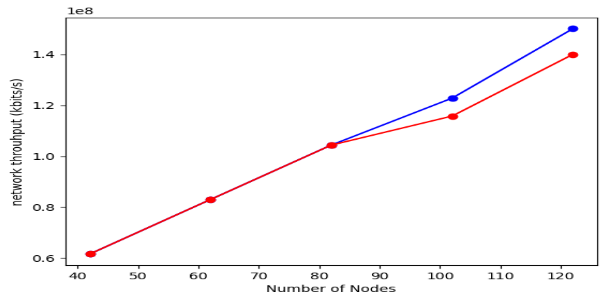
From figure 5, we see that TCP-Vegas and DC-Vegas performed similaly in incast scenarios. Our assumption was that in these scenarios DC-Vegas would outperform TCP-Vegas in terms of throughput. In incast scenarios TCP-Vegas throughput suffers heavily. But in this simulation we saw no such phenomenon, which suggests that our approach to simulate incast scenario was not perfect.

B. *For Wireless Network*

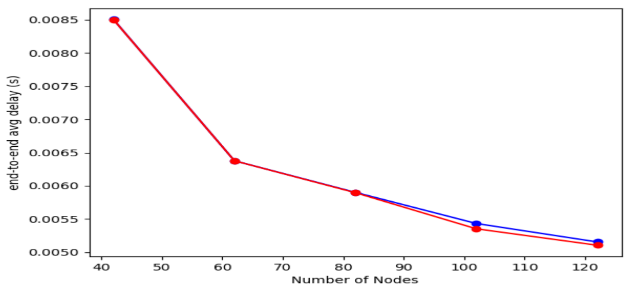
From figures 6, 7, 8, and 9, we can see that DC-Vegas almost always has a better throughput than TCP-Vegas, at the cost of higher delay rates. DC-Vegas also has higher packet delivery ratio and lower drop ratio. As the number of nodes increases, the throughput of both these two algorithms decrease while the end to end delay increase. On the contrary, as number of flows increases, both of these algorithms result in higher throughput and lower delays. In terms of energy, we find that the total energy consumption of these two algorithms are very close, but when DC-Vegas was used, nodes consumed less energy to transmit or receive a single packet of data on average. This result is consistent with the throughput results of these algorithms.

C. *Per-Node Metrics in Wireless Network*

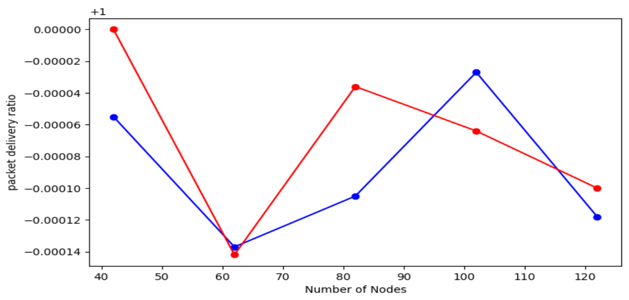
In this section, we attempt to compare per-node throughput, energy consumption and residual energy between DC-Vegas and TCP-Vegas while varying number of nodes and number of flows. In terms of per-node throughput, no significant pattern was found. But, per-node energy consumption and per-node residual energy graphs are very interesting. As shown in



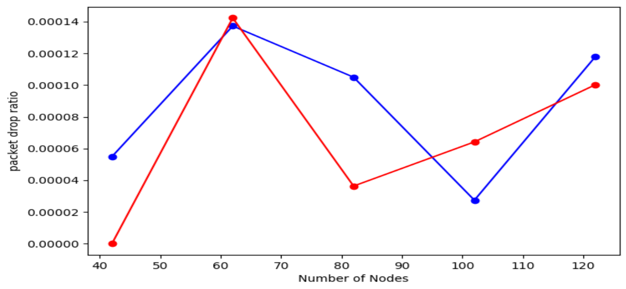
(a) Throughput vs Number of Nodes



(b) End-to-end Delay vs Number of Nodes

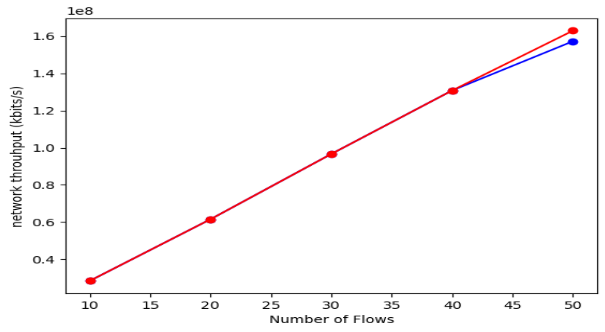


(c) Packet Delivery Ratio vs Number of Nodes

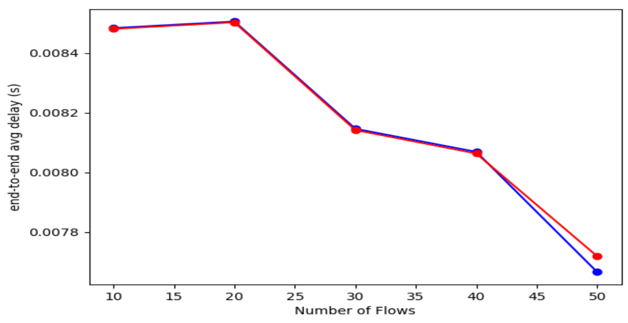


(d) Packet Drop Ratio vs Number of Nodes

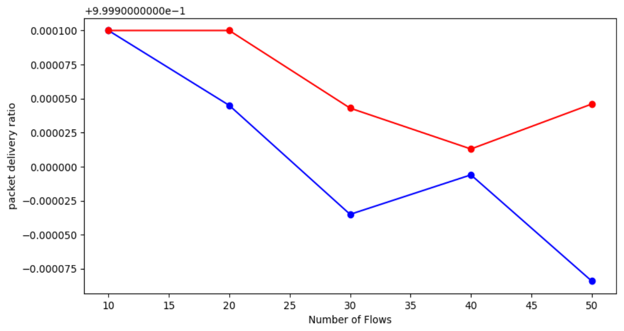
Fig. 2. **Affects of varying the number of nodes in wired network. Blue = TCP-vegas, Red = DC-Vegas**



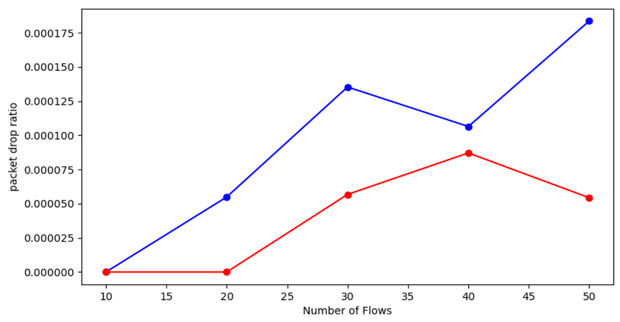
(a) Throughput vs Number of Flows



(b) End-to-end Delay vs Number of Flows

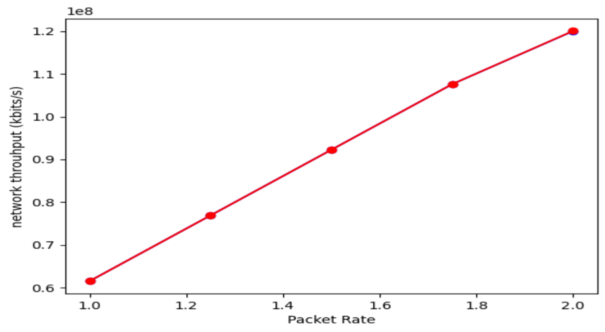


(c) Packet Delivery Ratio vs Number of Flows

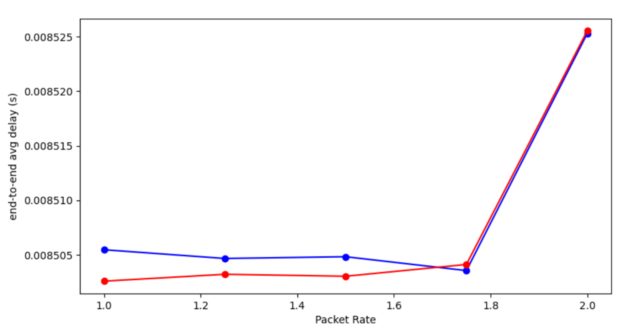


(d) Packet Drop Ratio vs Number of Flows

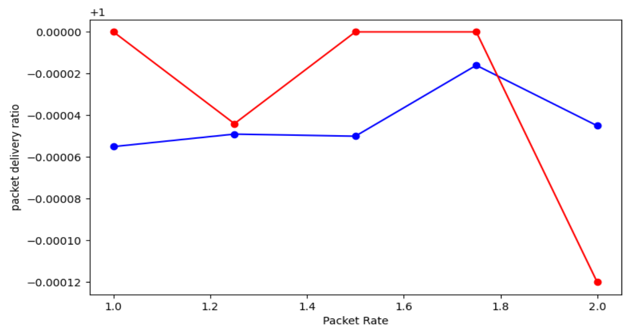
Fig. 3. **Affects of varying the number of flows in wired network. Blue = TCP-vegas, Red = DC-Vegas**



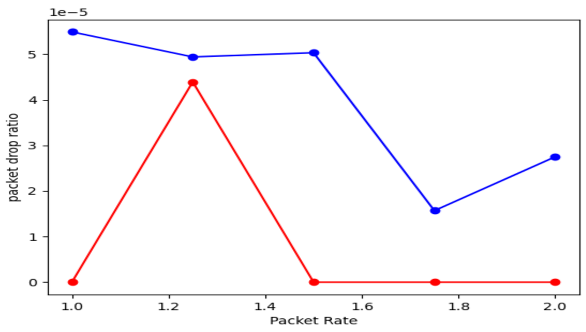
(a) Throughput vs Packet rate



(b) End-to-end Delay vs Packet rate

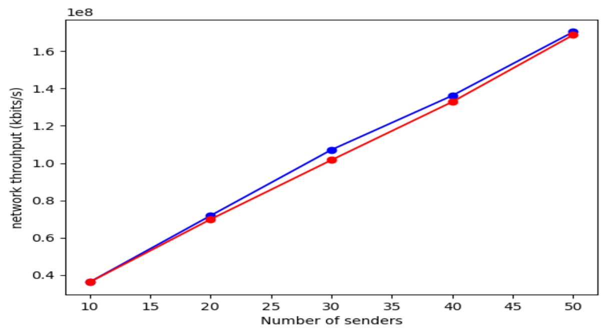


(c) Packet Delivery Ratio vs Packet rate

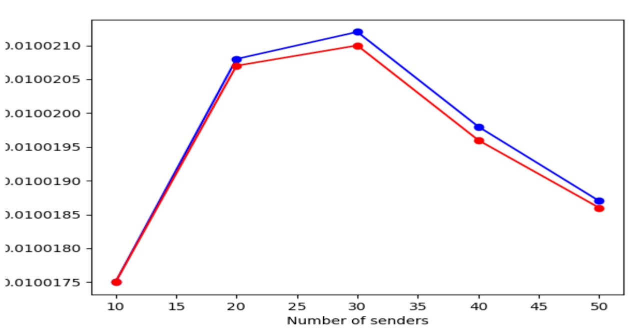


(d) Packet Drop Ratio vs Packet rate

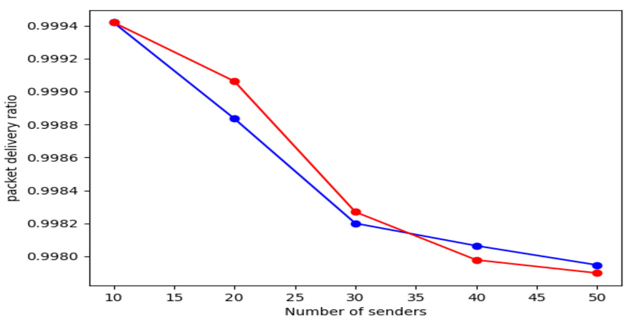
Fig. 4. **Affects of varying packet rate in wired network. Blue = TCP-vegas, Red = DC-Vegas**



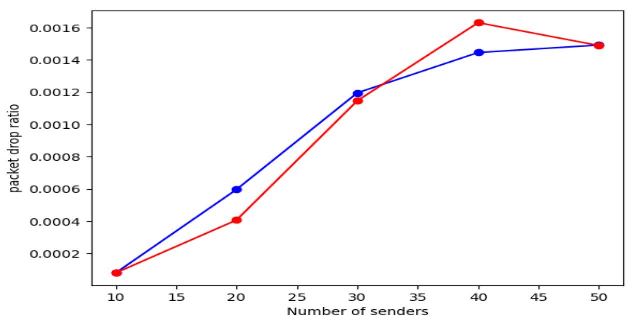
(a) Throughput vs number of senders



(b) End-to-end Delay vs number of senders

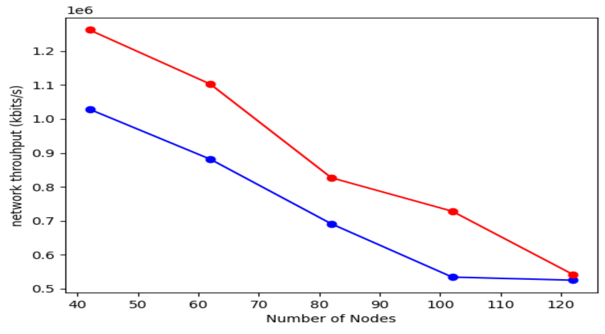


(c) Packet Delivery Ratio vs number of senders

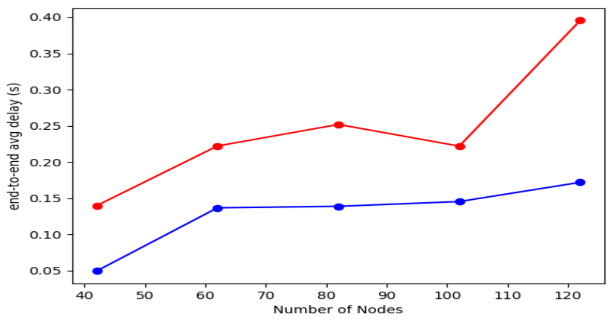


(d) Packet Drop Ratio vs number of senders

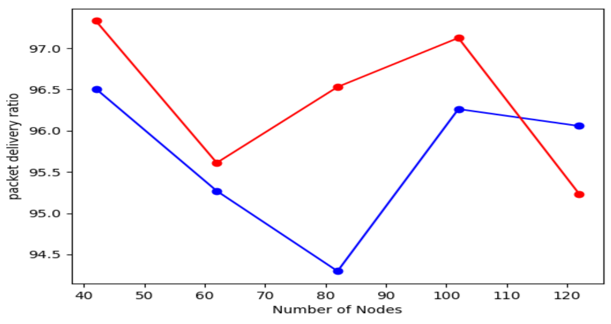
Fig. 5. **Affects of incast in wired network. Blue = TCP-vegas, Red = DC-Vegas**



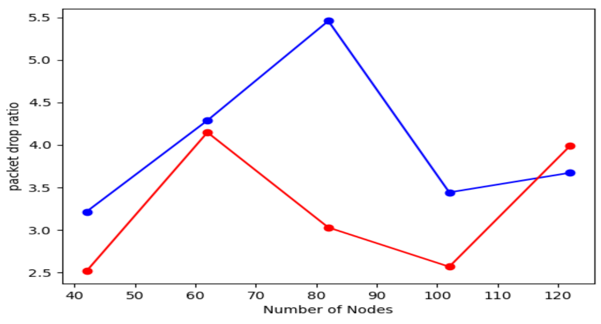
(a) Throughput vs Number of nodes



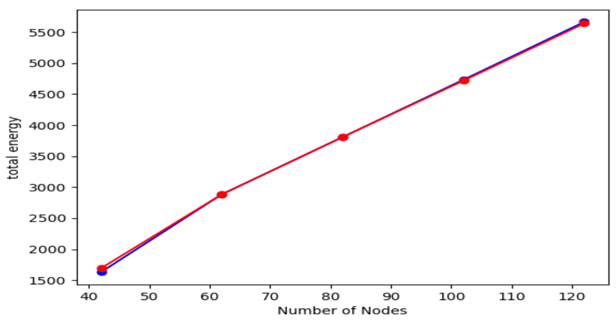
(b) End-to-end Delay vs Number of nodes



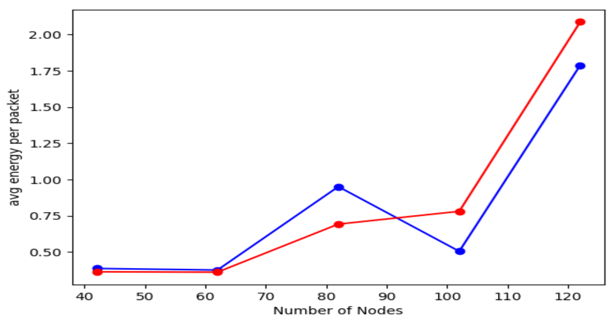
(c) Packet Delivery Ratio vs Number of nodes



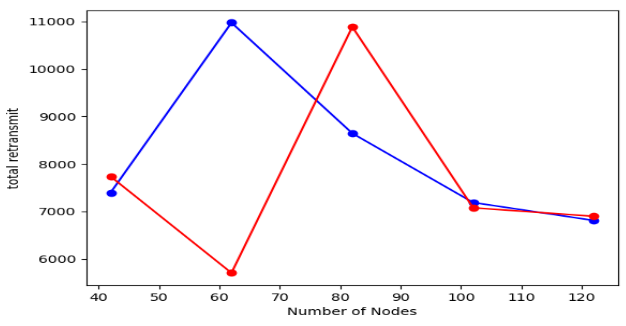
(d) Packet Drop Ratio vs Number of nodes



(e) Total Energy Consumed vs Number of nodes

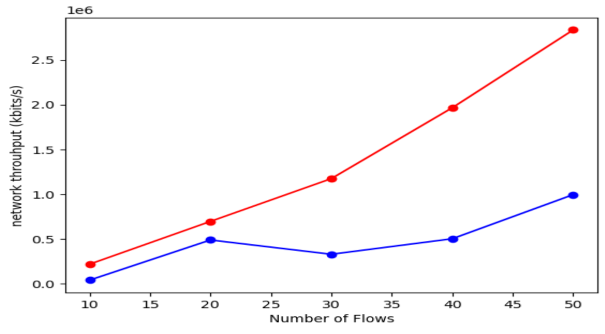


(f) Average energy consumed per packet vs Number of nodes

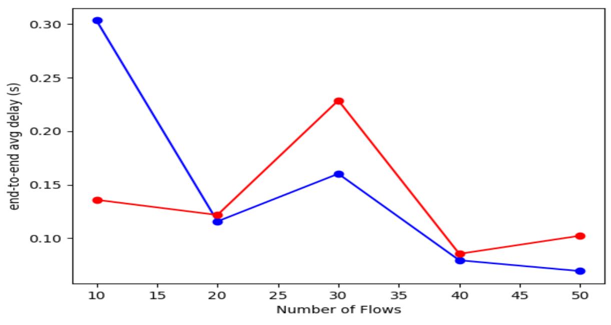


(g) Total Re-transmissions vs Number of nodes

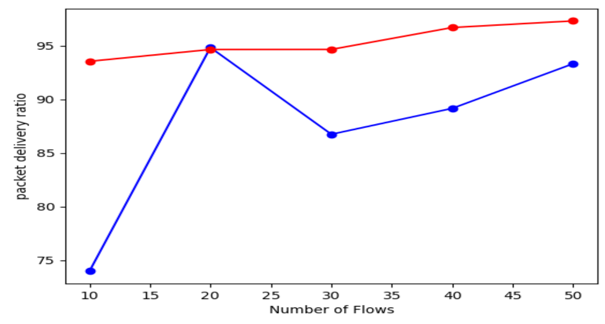
Fig. 6. **Affects of varying number of nodes in wireless network. Blue = TCP-vegas, Red = DC-Vegas**



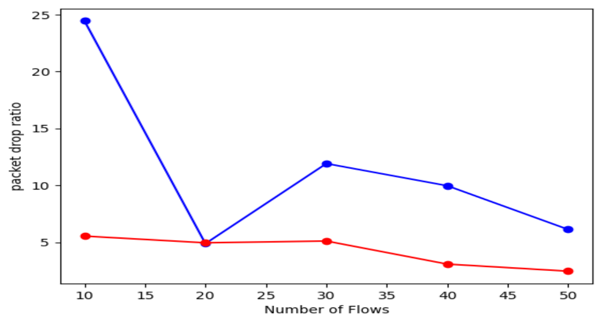
(a) Throughput vs Number of flows



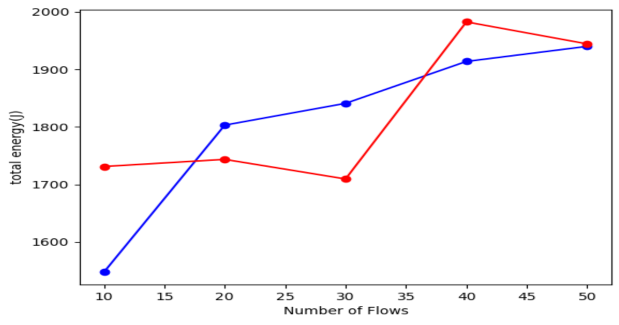
(b) End-to-end Delay vs Number of flows



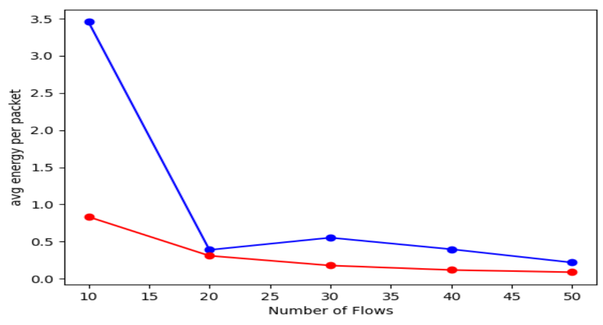
(c) Packet Delivery Ratio vs Number of flows



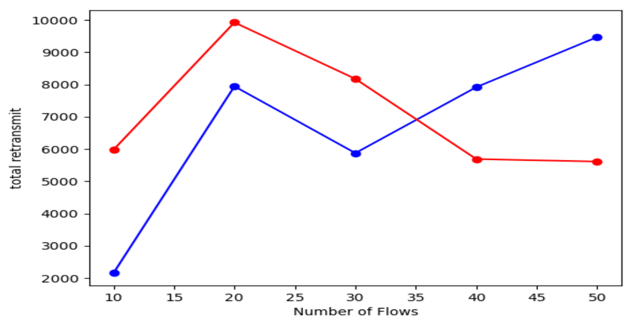
(d) Packet Drop Ratio vs Number of flows



(e) Total Energy Consumed vs Number of flows

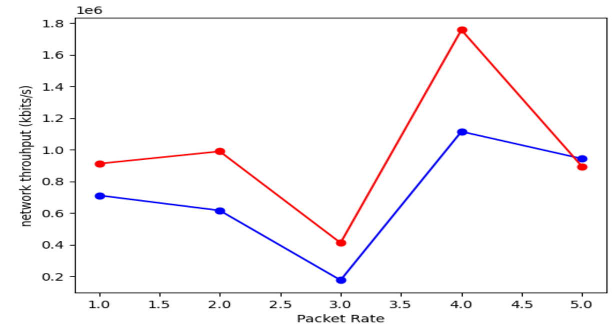


(f) Average energy consumed per packet vs Number of flows

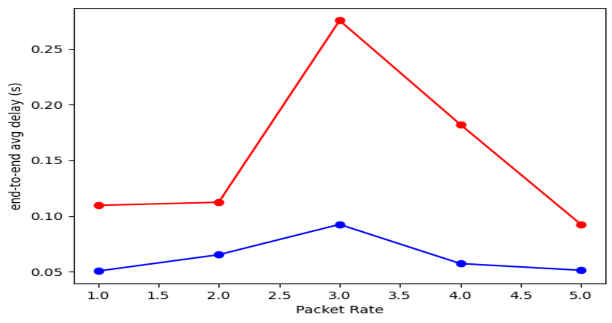


(g) Total Re-transmissions vs Number of flows

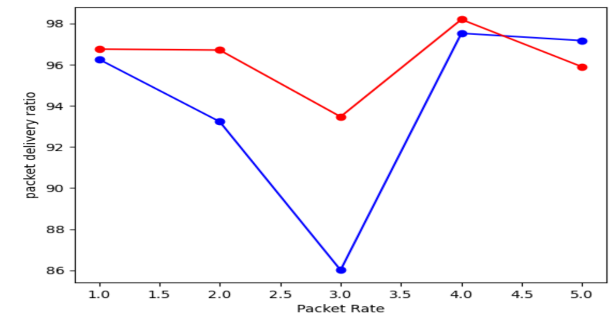
Fig. 7. **Affects of varying number of flows in wireless network. Blue = TCP-vegas, Red = DC-Vegas**



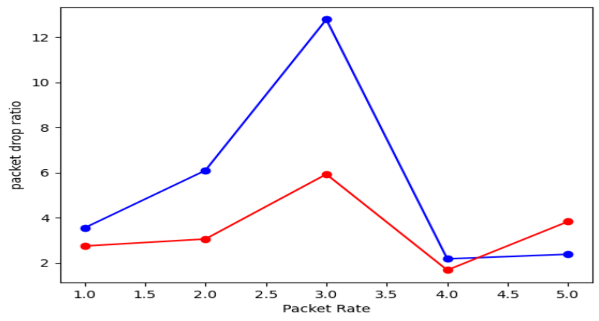
(a) Throughput vs Packet Rate



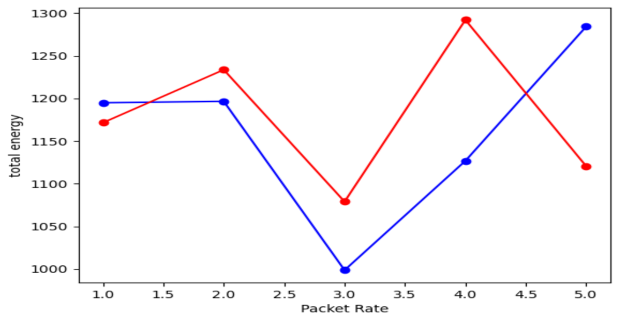
(b) End-to-end Delay vs Packet Rate



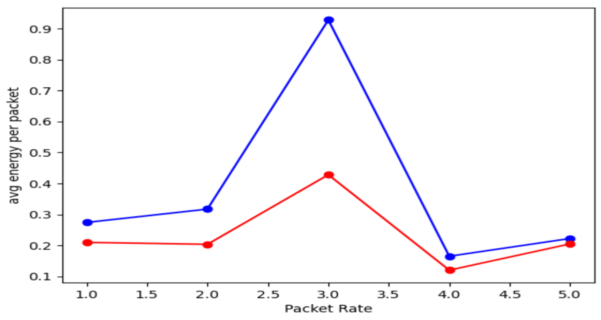
(c) Packet Delivery Ratio vs Packet Rate



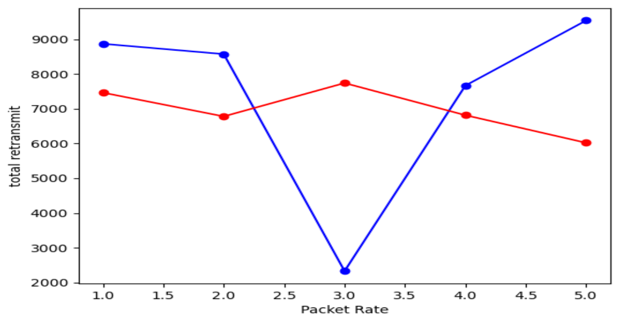
(d) Packet Drop Ratio vs Packet Rate



(e) Total Energy Consumed vs Packet Rate

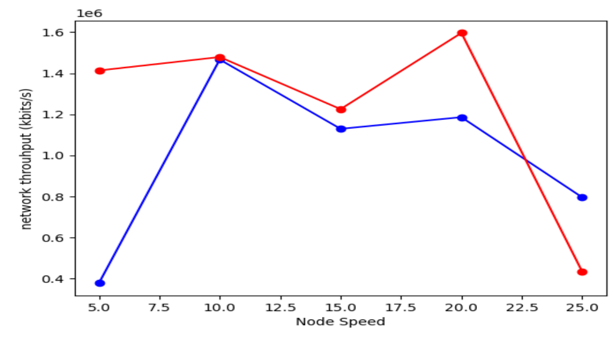


(f) Average energy consumed per packet vs Packet Rate

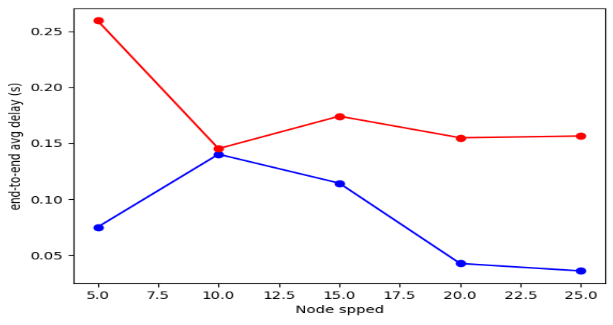


(g) Total Re-transmissions vs Packet Rate

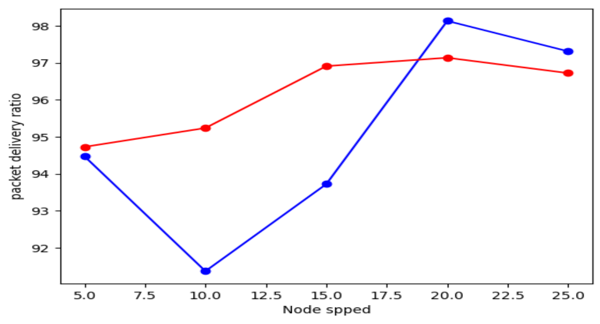
Fig. 8. **Affects of varying Packet Rate in wireless network. Blue = TCP-vegas, Red = DC-Vegas**



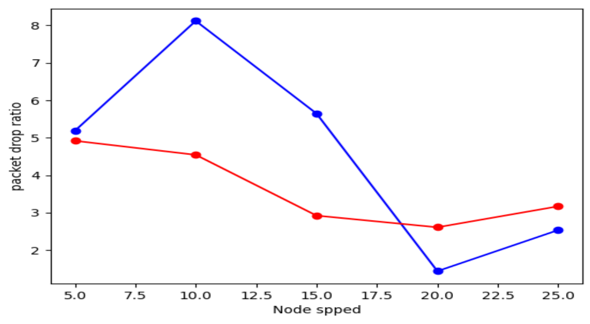
(a) Throughput vs Node Speed



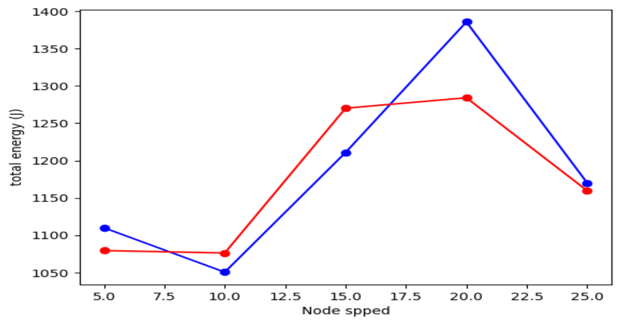
(b) End-to-end Delay vs Node Speed



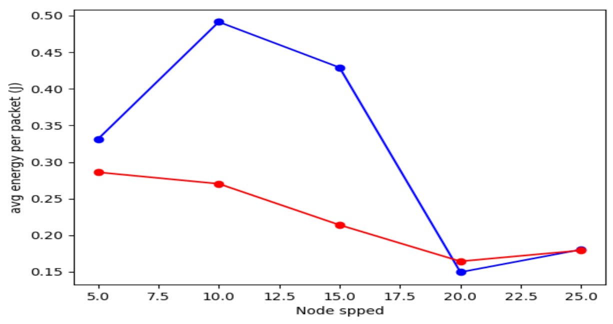
(c) Packet Delivery Ratio vs Node Speed



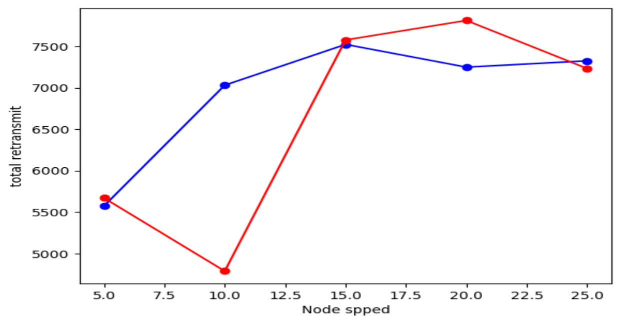
(d) Packet Drop Ratio vs Node Speed



(e) Total Energy Consumed vs Node Speed



(f) Average energy consumed per packet vs Node Speed



(g) Total Re-transmissions vs Node Speed

Fig. 9. **Affects of varying node speed in wireless network. Blue = TCP-vegas, Red = DC-Vegas**

figure 12 and 13, per-node energy consumption and residual residual energy stays roughly the same as the number of nodes increases. However, as we can see from figure 14 and 15, they vary greatly when number of flows increase.

VII. Experiment Limitations

There were some limitations while conducting this experiment, as mentioned below:

- We would have got a clearer result if we could conduct this experiment in a test-bed. As we found out during this experiment, it is very difficult to create incast scenarios in a simulation.
- Incast scenarios depend on the link bandwidth and delay time. The values used in this simulation may have not reflected that of an actual datacenter.

VIII. Summary Findings

Following is a summary of the findings from this experiment:

In wired network:

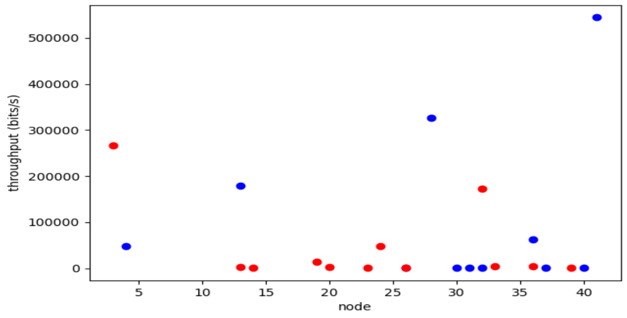
- As the number of flows and packet rate increase, DC-Vegas performs better than TCP-Vegas; with better packet delivery ratio and similar throughput and delay.
- As the number of nodes increase, DC-Vegas and TCP-Vegas perform roughly equal.

In wireless network:

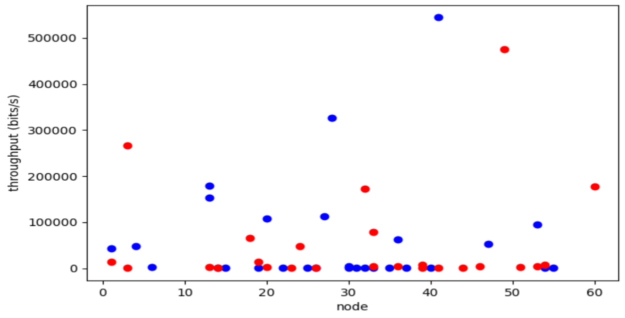
- DC-Vegas almost always have better throughput and better packet delivery ratio than TCP-Vegas.
- TCP-Vegas has lower end-to-delay than DC-Vegas.
- DC-Vegas graphs appear to be more consistent.

IX. Conclusion

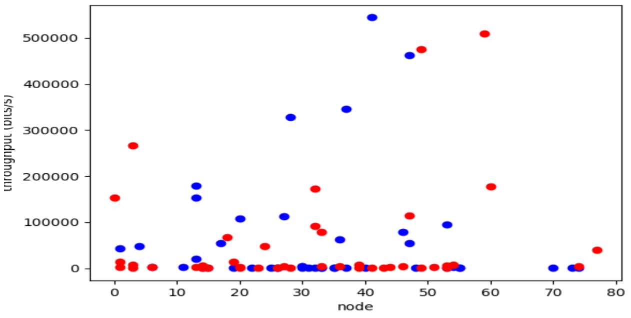
In this paper, DC-Vegas, a new delay-based TCP algorithm for datacenter congestion control, is proposed. DC-Vegas can achieve a performance comparable to state-of-the-art datacenter TCP algorithms and requires only minimal system modifications, i.e., only the update of TCP senders. Receiver side modifications, ECN support, and/or updates to other parts of datacenter networks are unnecessary, which is very valuable for existing production data centers. Experiments conducted an ns-2 simulations confirmed the performance (including throughput, end-to-end delay, packet delivery and drop ratio) of the proposed algorithm.



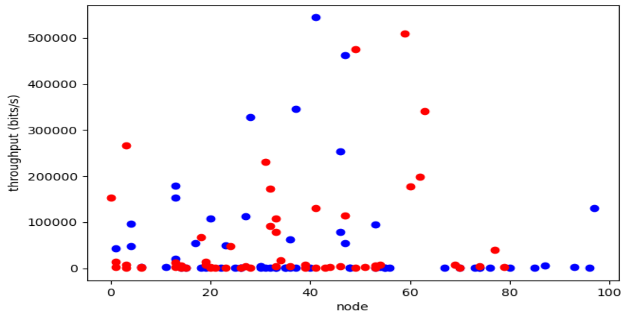
(a) For 42 nodes



(b) For 62 nodes

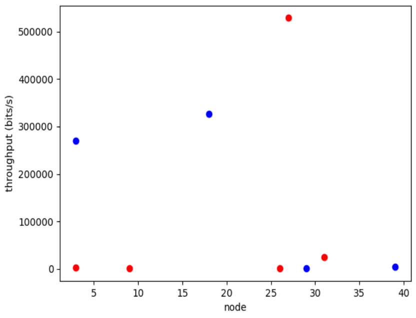


(c) For 82 nodes

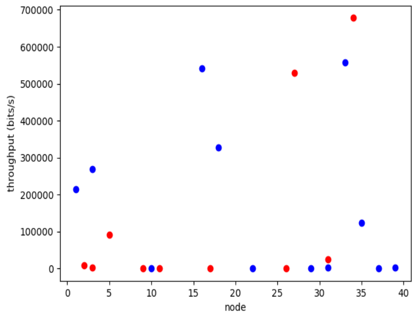


(d) For 102 nodes

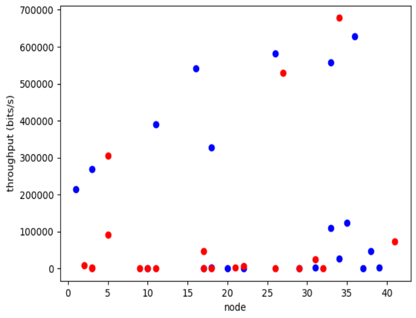
Fig. 10. **Per-node throughput for varying number of nodes in wireless network (Number of flows: 20). Blue = TCP-vegas, Red = DC-Vegas**



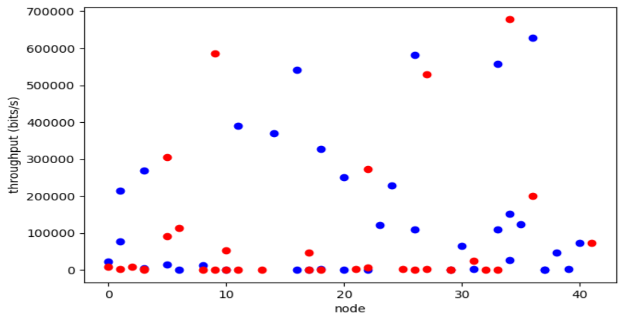
(a) For 10 flows



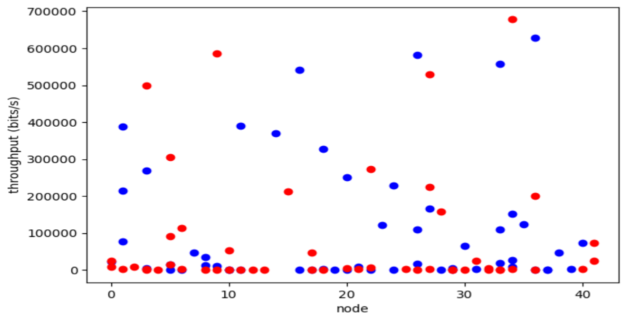
(b) For 20 flows



(c) For 30 flows

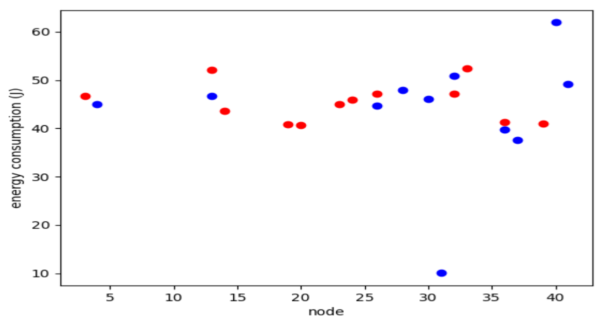


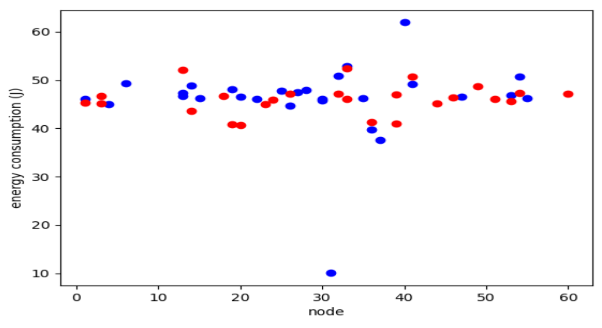
(d) For 40 flows

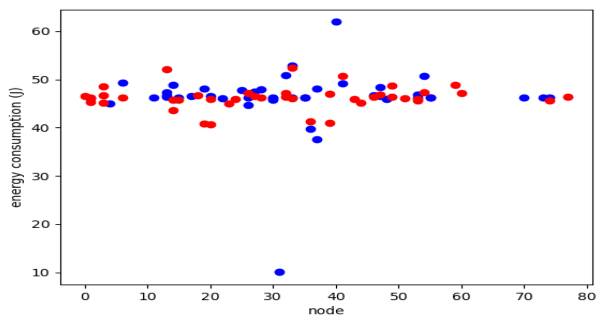


(e) For 50 flows

Fig. 11. **Per-node throughput for varying number of flows in wireless network (Number of nodes: 42). Blue = TCP-vegas, Red = DC-Vegas**


(a) For 42 nodes


(b) For 62 nodes


(c) For 82 nodes

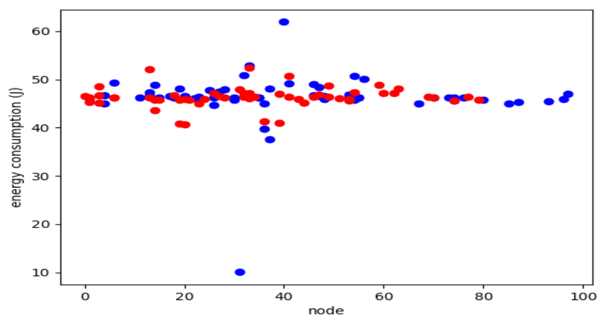
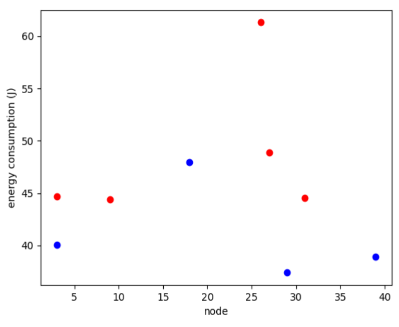
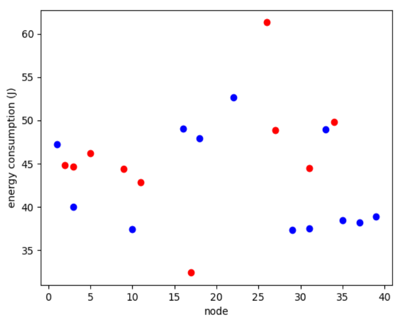
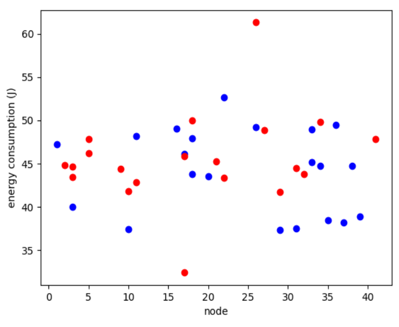
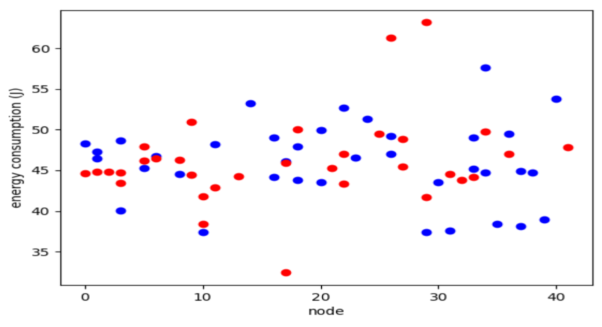

(d) For 102 nodes

Fig. 12. **Per-node enrgy consumption for varying number of nodes in wireless network (Number of flows: 20). Blue = TCP-vegas, Red = DC-Vegas**


(a) For 10 flows


(b) For 20 flows


(c) For 30 flows


(d) For 40 flows

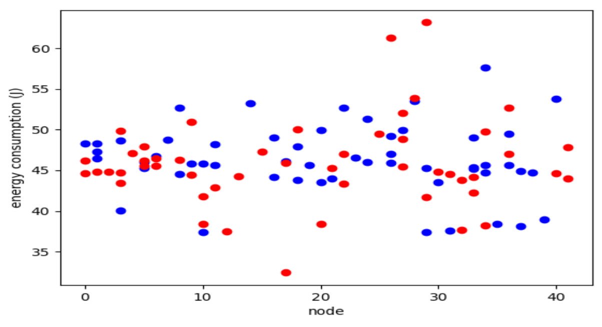
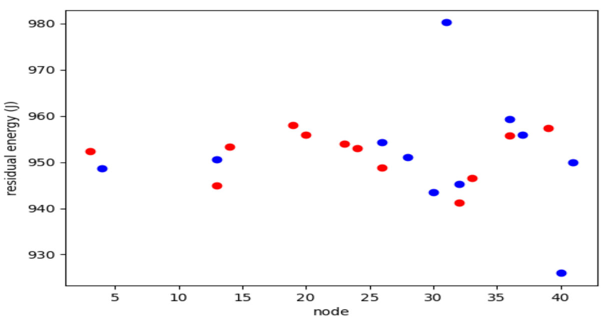
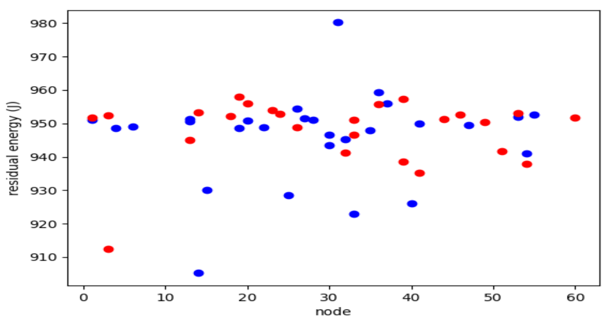

(e) For 50 flows

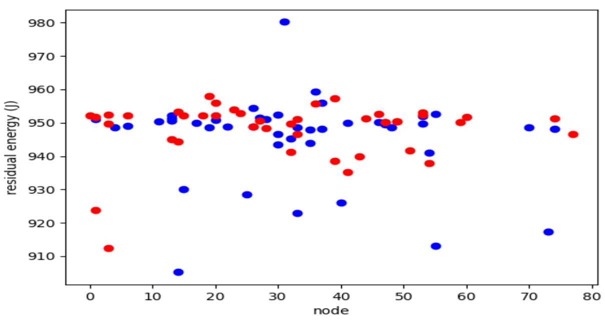
Fig. 13. **Per-node energy consumption for varying number of flows in wireless network (Number of nodes: 42). Blue = TCP-vegas, Red = DC-Vegas**



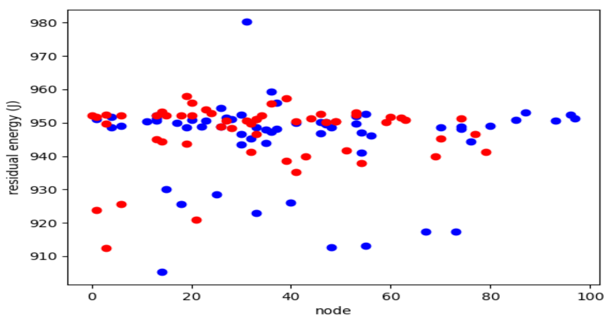
(a) For 42 nodes



(b) For 62 nodes

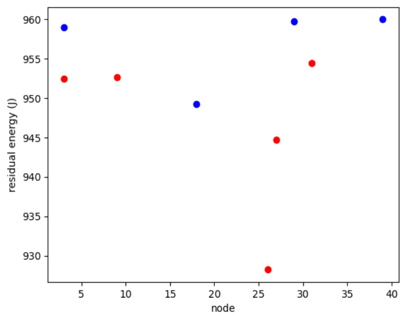


(c) For 82 nodes

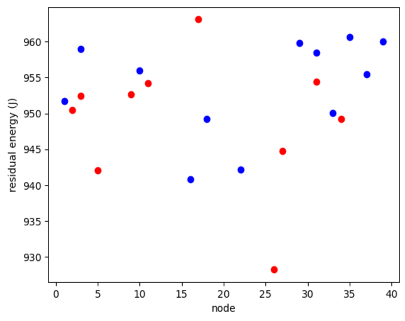


(d) For 102 nodes

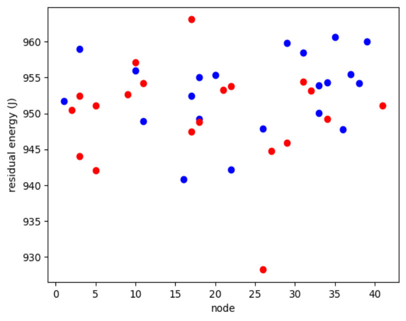
Fig. 14. **Per-node residual energy for varying number of nodes (Number of flows: 20) in wireless network. Blue = TCP-vegas, Red = DC-Vegas**



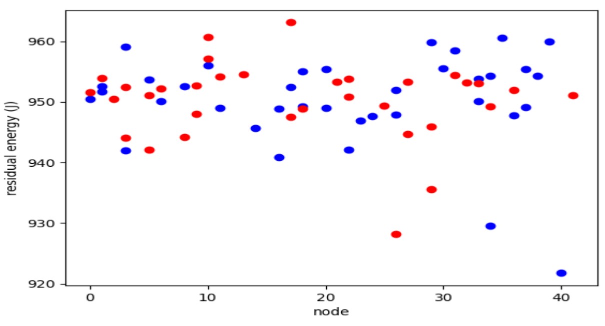
(a) For 10 flows



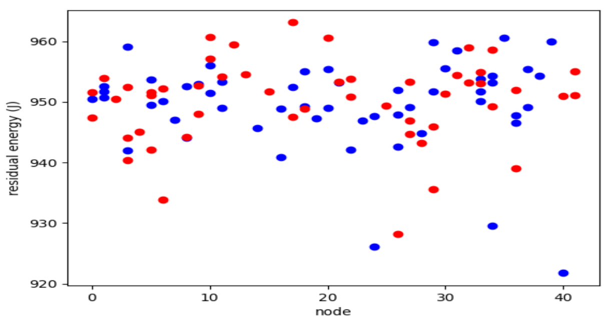
(b) For 20 flows



(c) For 30 flows



(d) For 40 flows



(e) For 50 flows

Fig. 15. **Per-node residual energy for varying number of flows (Number of nodes: 42) in wireless network. Blue = TCP-vegas, Red = DC-Vegas**