Profiling Network Components Using Keddah

Jie Deng
School of Electronic Engineering and
Computer Science
Queen Mary University of London
Email: j.deng@qmul.ac.uk

After successfully reproduced the Hadoop traffic in the simulation, we can stack more functions upon the traffic to evaluate the network elements such as topology, queue, routing protocols or congestion control. In this section, we shown how the workload generator could be used for different scenarios such as data center topology evaluation, network components, TCP efficiency and cluster size.

The workload we generated consists of equal number of both TeraSort 3G and kmeans with 50 cluster and 50k samples jobs, in a random sequence. The job parameters can be set differently for better flexibility, but we use a fixed job setup for testing purpose as the traffic various in the same job already. We adjusted the load by select different parameters for the exponential distributed job submitting interval, to help expose the network behaviours.

TABLE I: Experiment traffic model parameters.

Name	Decscription	Value
n	cluster size	$\begin{cases} FatTree: & 64 \\ CamCube: & 64 \\ DCell: & 42 \end{cases}$
b	mapred.max.split.size	128MB
k	convergence factor	4
s	TeraSort file size	3GB
clusters	Number of kmeans clusters	50
samples	Number of kmeans samples	50k

A. Network Topologies

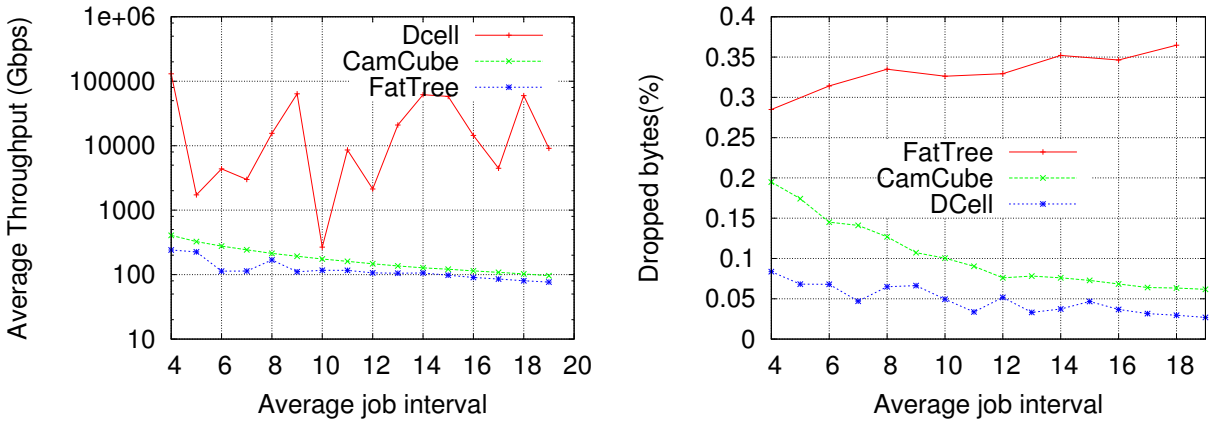
Previously people have study the data center topologies [**?**] to see how the network evolves with the scale of data center. The data center network topology design will involve in various challenges such as cost, performance, reliability, scalability, security and energy. Work degree, throughput [8] compared the theoretical all to all traffic matrix throughput to evaluate the throughput of different topologies. Also work [6] discussed the reliability of network topologies by network size and path stretch rate. It shows hybrid topologies like BCube and DCell degrades smoother than Fat-Tree in terms of link and switch failures, however with a cost of substantially.

Moreover, work [3] has proposed to use ns3 to simulate the network topologies and compare the throughput and delay under uniform or exponential distributed traffic. Interestingly, the simulated throughput performance of FatTree and DCell are not match with theoretical value shown in work [8], which could be caused by the simulated workload. That said, the workload generated by different application could affect the performance of network topologies. The traffic may have different requirements for the network to congest, such as latency, throughput or packet loss. Here we reproduce Hadoop specific traffic, can help us explore the performance of different network topologies, with a more convincing traffic workload.

We proposed a test for different topologies under the Hadoop traffic workload. The topologies include FatTree [1], Dcell [7] and Camcube [5] as these three are very typical network topology represent for clos, star and torus. We implemented each topology in ns3 using “Node” as the computation nodes and “PointToPointHelper” or “CsmaHelper” as the link in between. The routing is handled by “GlobalRouteManager” with ECMP(equal-cost multi-path routing) enabled. The topology instance we use in the experiment is FatTree(8x8), CamCube(4) and DCell(2,2) which contains 64, 64 and 42 nodes respectively.

The throughput result shown in our simulation match the result of work [8] in terms of the all-to-all or random selected traffic matrix. Though not like [3] the throughput of Dcell drop below FatTree after 32 nodes, the Dcell throughput did varies a lot for different traffic load due to topologies structure.

Moreover, as shown in Fig. **??**, we compared the packet congestion caused loss of different topologies under different job loads which no other works have investigated. We can see that the DCell has the lowest drop rate and still decreasing over reduced traffic loads. However, the FatTree not only got the highest drop rate, and the rate is increasing over the traffic load.



(a) (b)

Fig. 1: (a) Compare the average throughput of three topologies in different workload. The same result can be found from previous work. (b) Compare the packet dropped in three different topologies.

B. TCP incast

TCP incast has widely studied as it is considered as a serious performance killer in data center [4], and MapReduce used in Hadoop is a vulnerable application as many-to-one transmission pattern is happen often. Different approaches have been addressed for reduce the congestion, such as using fine-grained TCP retransmission [10] or DCTCP(Data Center TCP) which using ECN(Explicit Congestion Notification) for congestion notification [2]. However, few work have really focus on Hadoop system expected [**?**] found smaller TCP RTO(ReTransmission timeOuts) can reduce the running time of hdfsread, hdfswrite and shuffle operations. Nothing has touched the job or workload perspective of Hadoop system.

We use the same workload consists of TeraSort and kmeans jobs to explore the TCP performance under different loads, and explicitly export the TCP RTT(round-trip time), RTO and transmission windows in ns3 during data transmission. We found, the transmission window is highly correlated to the congestion as shown in Fig 3. Each time a packet dropped, the window size will decreasing as well.

Transmission window can reflects the TCP performance for one session, we calculate the concurrent transmission window decreasing to capture TCP incast. Also, we run the workload with different parameters for the exponential job interval to adjust the traffic load. We found

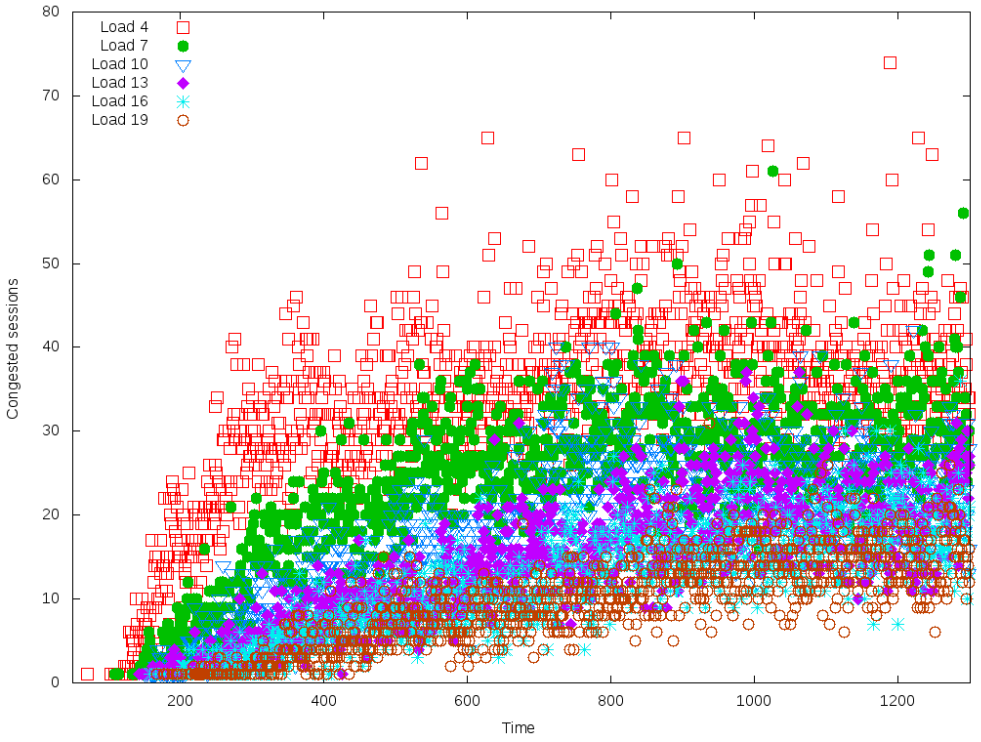


Fig. 2: .

indeed, the higher traffic load has more concurrent TCP window decreasing events over time, seems seriously damage the TCP performance.

However, when look at the throughput of each workload as in Fig. 4, we found higher congestion not necessary means worse performance. The throughput of higher traffic load is higher too, in spite of the congestion. That said, the workload is highly related to TCP performance tuning, as well as the hardware capacity. Before apply any complicate congestion control algorithm to improve one session, one can simply increase the traffic load to leverage a better global performance and hardware utilization.

C. TCP vs DCTCP

Beyond that, we can also compare the performance of TCP and DCTCP for Hadoop traffic. Not like the traffic patterns evaluated in its original design, the Hadoop workload is full of short flows with high burstiness. We use DCTCP implementation in ns3 [1], and RED queue with

[1]https://github.com/i-maravic/ns-3

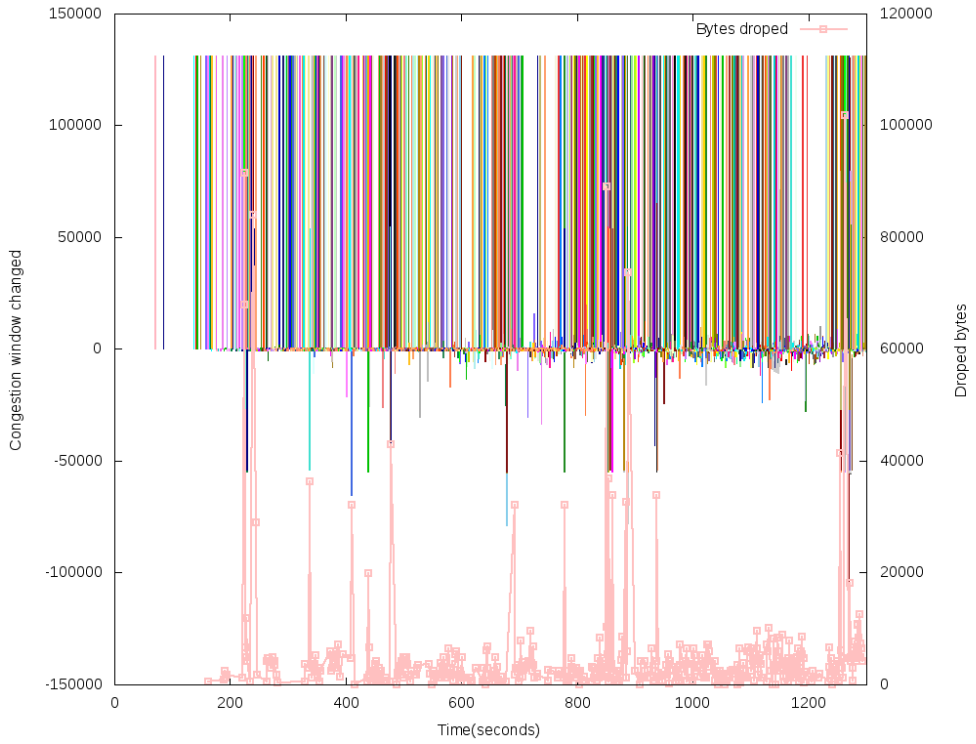


Fig. 3: Transmission window value changed and the number of bytes dropped at time x. The packet drop always causing window size decreasing.

marking threshold at respectively 100 and 40 packets. Using the Hadoop workload consists of TeraSort and Kmeans jobs, running on the DCell(2,3) topology. We found the DCTCP isn't always improve the performance in terms of throughput and packet loss. More precisely, as shown in Fig 5a and 5b, the DCTCP offer the same packet loss with lower throughput in high traffic load, and lower packet loss with the same throughput in low traffic load. That said, due to the difference in traffic pattern, the DCTCP does not improve the performance dramatically.

D. Link vs Bandwidth

Different topologies will congest the traffic in different ways, thus performance may various. We saw how the DCell and CamCube are achieving a better performance then FatTree in the Hadoop scenario. Intuitively, the DCell and CamCube have more node to node links where as in FatTree nodes are only connected to the access switch. Work [11] has discussed the property of data center network topologies like number of paths and shortest path when compare the

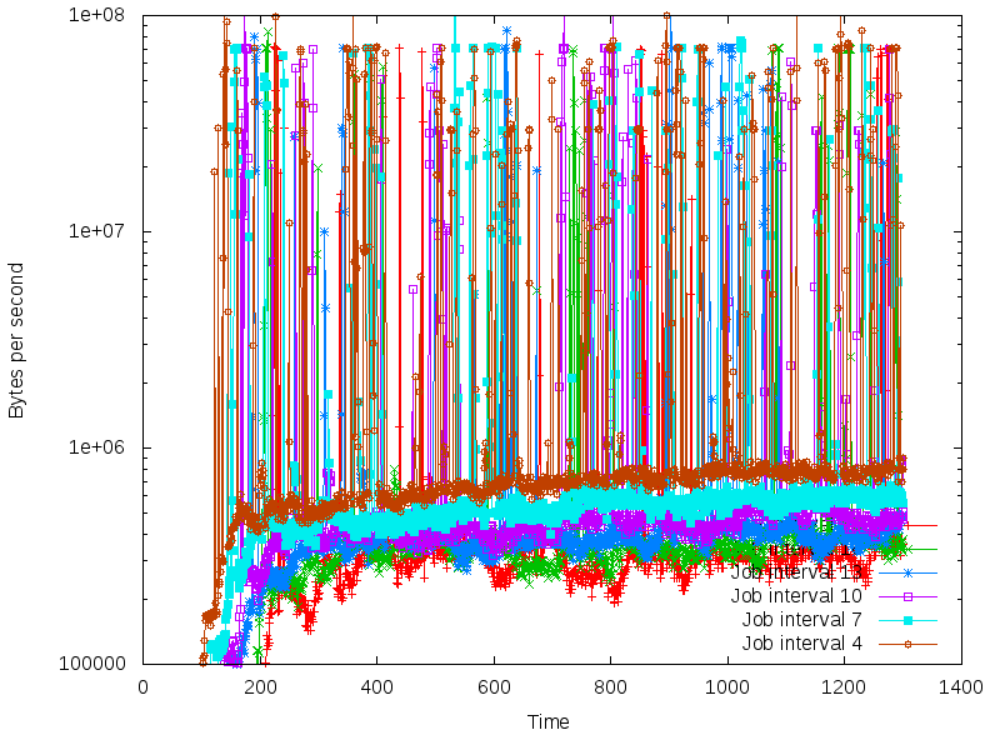
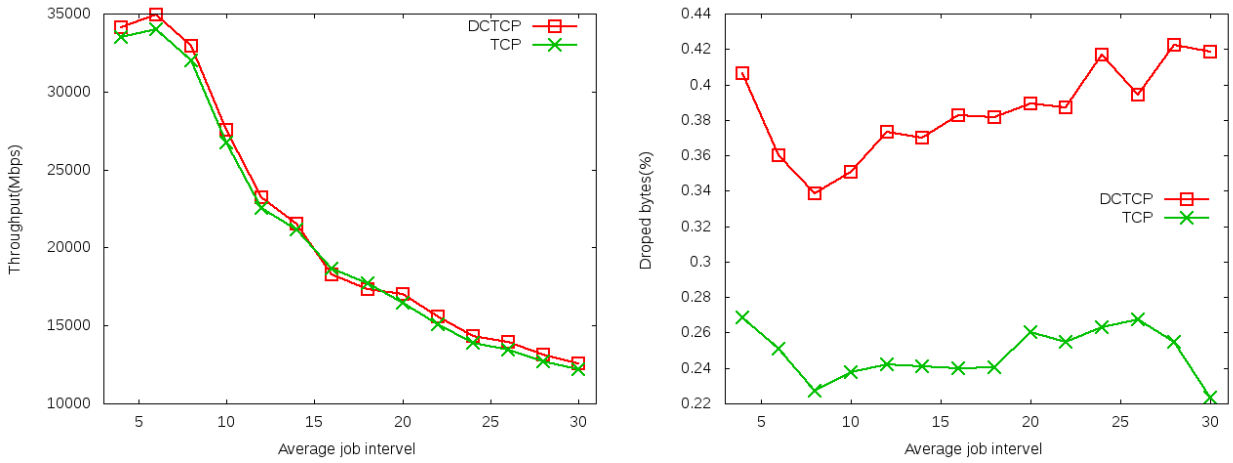


Fig. 4

performance. Here, we can use a more participial approach evaluate the how important the interlink between nodes is.

We select the CamCube as our test topology as all the links in CamCube are node to node and each node share 3 links with other nodes, not like nodes in FatTree and DCell only have one up-link. As remove one each link reduces 33% of the total link, and we want to explore it with smaller granularity to explore the performance. We set a interlink factor k from 100 to 50 by a step of 5 to randomly reduce the link between nodes. When the $k = 90$, the connectivity of nodes will be 90% of the original CamCube, which means 10% of the link between nodes are purged by randomly. To keep to transmission capacity, we also increase the bandwidth of the connected links by $1 - k$, so we can compare which is more important for the transmission: link or bandwidth.

By using the Hadoop specific workload onto different settings, we found the relationship between link and bandwidth is not simply one thing is more important than the other. Though the throughput of different settings seems to be the same, but the packet drop may various a lot! As shown in Fig. 6, the drop rate is actually the highest in the original CamCube setting, and



(a) Throughput, higher the better.

(b) Packet loss, lower the better.

Fig. 5: TCP vs DCTCP using Hadoop workload.

decreasing until reach the bottom when $k = 0.8$. After that, the drop increased slowly and stand still after $k = 0.7$. That said, though the traffic pattern in the cluster is multi-point to multi-point, a trade off between interlink and bandwidth can be investigated. Apart from the interlink and bandwidth, the queue or other factors may effect the performance. Only the simulation process can provide opportunities to explore the trade off.

E. Minimum cluster size

Though the right cluster size will depends on the processing ability of the working nodes, we can take a look at the network scale for the certain workload. Work [9] have studied the cost of different data center network topologies in terms of network devices and network components required. Here, we can dynamically increasing the network scale until the performance fulfill the workload in the simulate approach.

We use FatTree as our test topologies as the nodes in FatTree can be increasing by a smaller step then DCell and CamCube. The number of pods has set to 8, and we increasing of number of nodes per pod from 4 to 20. We can see that, as expected the drop rate is linearly decreasing over the increased number of computing nodes from 1.4% to 0.2%, and keep at 0.2% afterwards. To further lower the drop rate requires higher bandwidth between each nodes, thus the cost will be much more than a single PC.

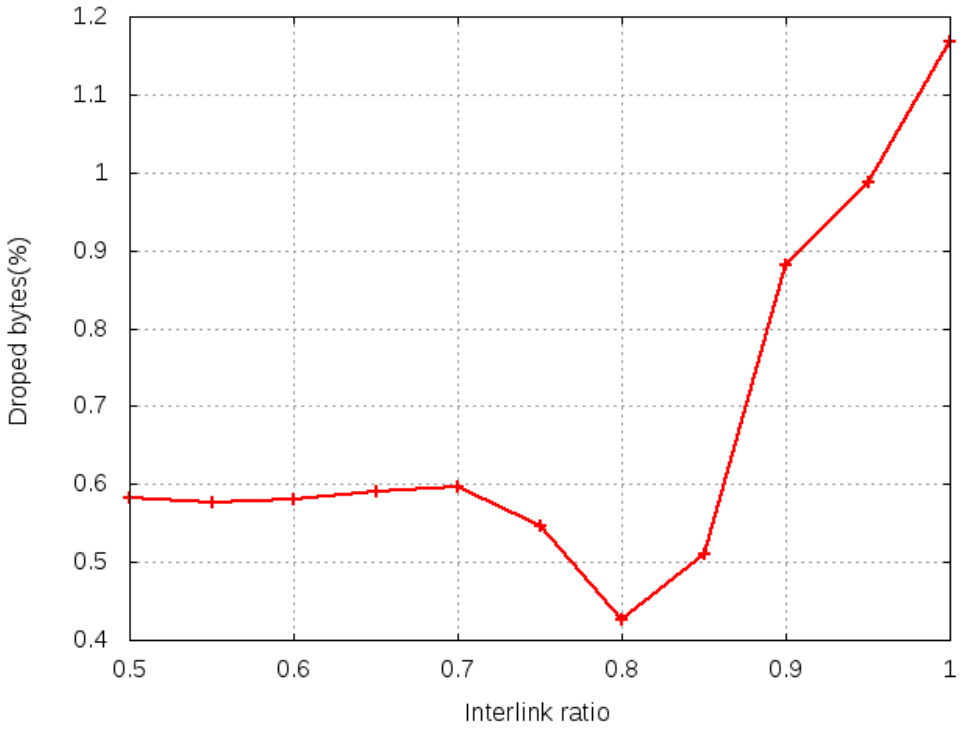


Fig. 6: The trade off between interlink and bandwidth valued by bytes dropped. Tunning the topology parameters can help improve the network performance, and here we found the convergence point for Hadoop traffic.

This approach can be used for estimate the acceptable scale of a cluster, in terms of network performance and budget. After recording the computing performance of one machine, one can imitate the performance of that machine, as replay it in the simulation to infer the performance of a larger cluster.

REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *ACM SIGCOMM Computer Communication Review*, 38(4):63–74, 2008.

[2] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center tcp (dctcp). *ACM SIGCOMM computer communication review*, 41(4):63–74, 2011.

[3] K. Bilal, S. U. Khan, L. Zhang, H. Li, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, D. Chen, M. Iqbal, et al. Quantitative comparisons of the state-of-the-art data center architectures. *Concurrency and Computation: Practice and Experience*, 25(12):1771–1783, 2013.

[4] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph. Understanding tcp incast throughput collapse in datacenter networks. In *Proceedings of the 1st ACM workshop on Research on enterprise networking*, pages 73–82. ACM, 2009.

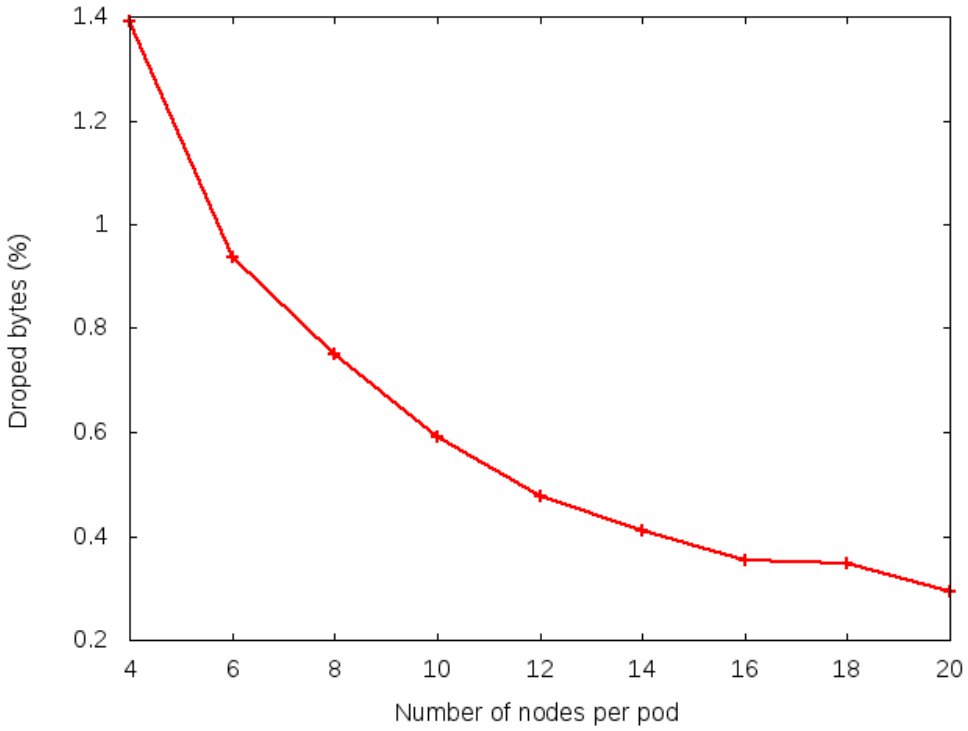


Fig. 7: Compare the packet dropped in FatTree topologies over increasing number of nodes per pod. A liner trend can be found in data bytes dropped over linearly increasing the number of nodes. We can estimate the processing nodes required for workload by keep increasing the nodes until reasonable packet drop.

[5] P. Costa, A. Donnelly, G. Oshea, and A. Rowstron. Camcube: a key-based data center. Technical report, Technical Report MSR TR-2010-74, Microsoft Research, 2010.

[6] R. S. Couto, M. E. M. Campista, and L. H. M. Costa. A reliability analysis of datacenter topologies. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 1890–1895. IEEE, 2012.

[7] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. Dcell: a scalable and fault-tolerant network structure for data centers. *ACM SIGCOMM Computer Communication Review*, 38(4):75–86, 2008.

[8] S. A. Jyothi, A. Singla, P. Godfrey, and A. Kolla. Measuring and understanding throughput of network topologies. *arXiv preprint arXiv:1402.2531*, 2014.

[9] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, and I. Stoica. A cost comparison of datacenter network architectures. In *Co-NEXT '10 Proceedings of the 6th International COnference*, page 16. ACM, 2010.

[10] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller. Safe and effective fine-grained tcp retransmissions for datacenter communication. In *ACM SIGCOMM computer communication review*, volume 39, pages 303–314. ACM, 2009.

[11] F. Yao, J. Wu, G. Venkataramani, and S. Subramaniam. A comparative analysis of data center network architectures. In *Communications (ICC), 2014 IEEE International Conference on*, pages 3106–3111. IEEE, 2014.