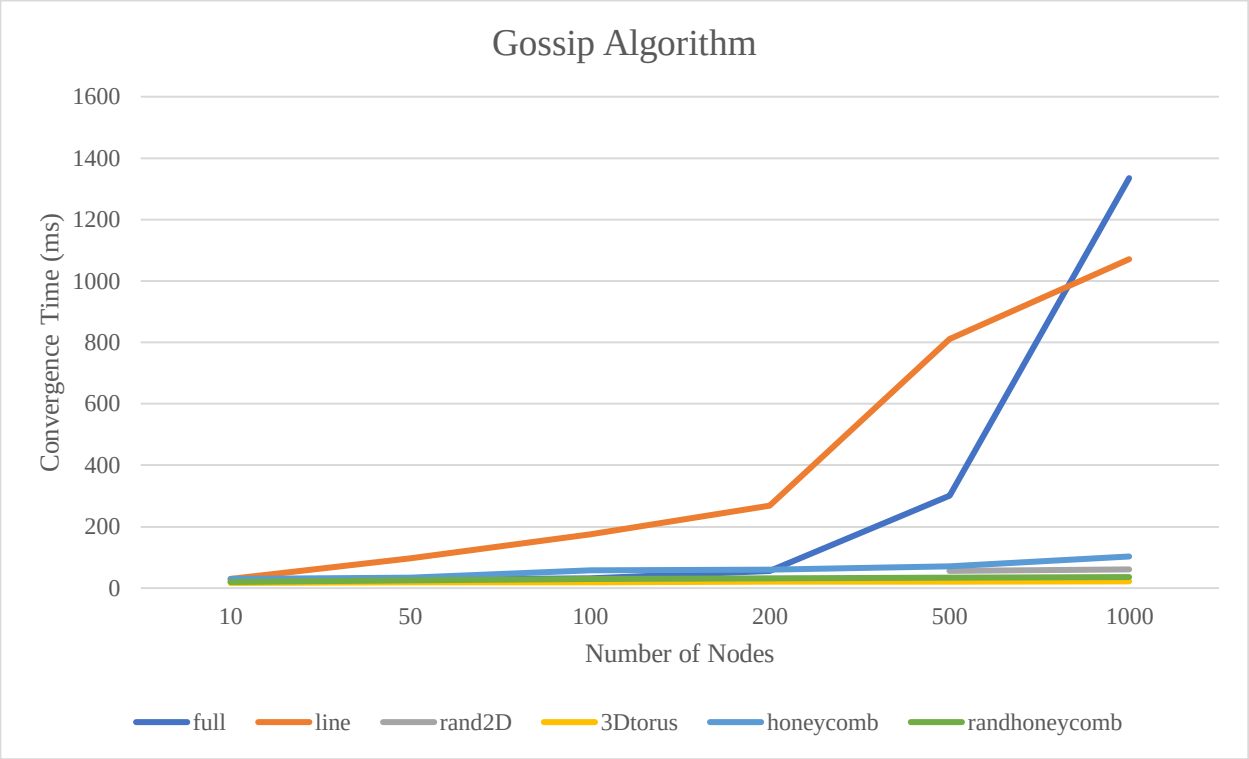
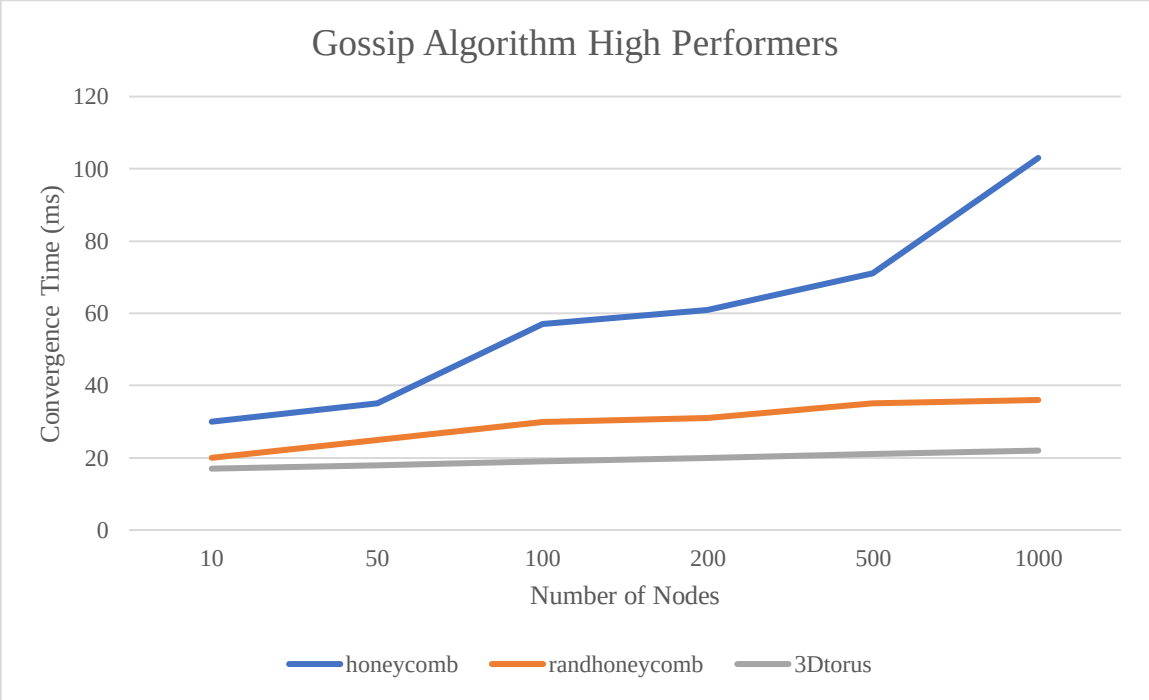
Kayton Fletcher – Report Project 2

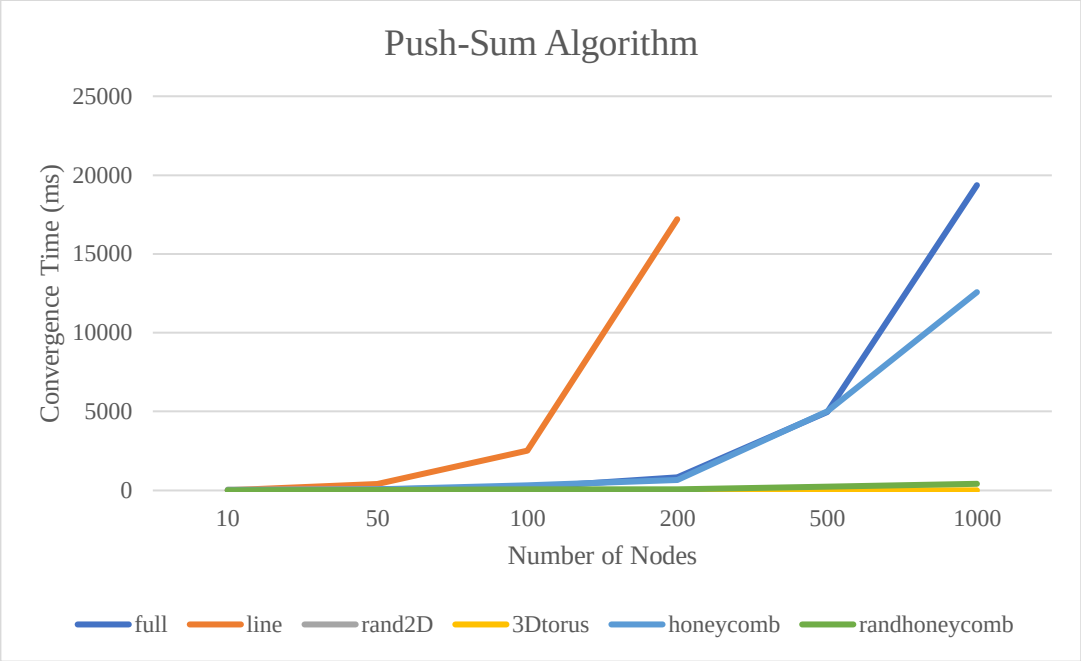




Kayton Fletcher – Report Project 2

For the gossip algorithm, it was easy to see that honeycomb, randhoneycomb, 3Dtorus and rand2D performed the best. Full easily performed the worst as the number of nodes increased, having far too much overhead to keep up with the message sending. Rand2D has the fault of not working with less than 400 nodes, but above that it performed surprisingly well. Line did not have the best performance, but as the number of nodes increased it handle the increase well, performing logarithmically as expected.

gossip						
	10	50	100	200	500	1000
full	22	28	32	55	300	1335
line	30	97	175	269	811	1071
rand2D					56	61
3Dtorus	17	18	19	20	21	22
honeycomb	30	35	57	61	71	103
randhoneyco	20	25	30	31	35	36



The push-sum algorithm exhibited many of the same traits that gossip had. Nearly across the board it took longer for push-sum than for gossip to complete, with one exception. 3Dtorus performed outstanding with the push-sum algorithm, handling 300000 nodes in just 35ms. It is no surprise that torus topologies of 5 and 6 dimensions are what some modern super computers use to compute heavy workloads. In push-sum, line actually performed worse than full. Full, line and honeycomb all got exponentially worse as the number of nodes increased. Although gossip algorithms are meant to show logarithmic convergence, this shows that many topologies make compromises that can make or break the performance of a distributed system.

push-sum						
	10	50	100	200	500	1000
full	7	52	182	824	4967	19362
line	12	414	2519	17201	0	0
rand2D	0	0	0	0	56	61
3Dtorus	7	7	7	7	7	8
honeycomb	16	58	315	676	5026	12576
randhoneyco	11	23	56	86	241	422