1 Evaluation

1.1 Bandwidth

Jocelyn ▶EpTO *was run with* $c = 4$ *to prevent any holes and a* δ *period of 100ms*◀ In Figure 1 we can see EpTO has a worse baseline compared to JGroups. It

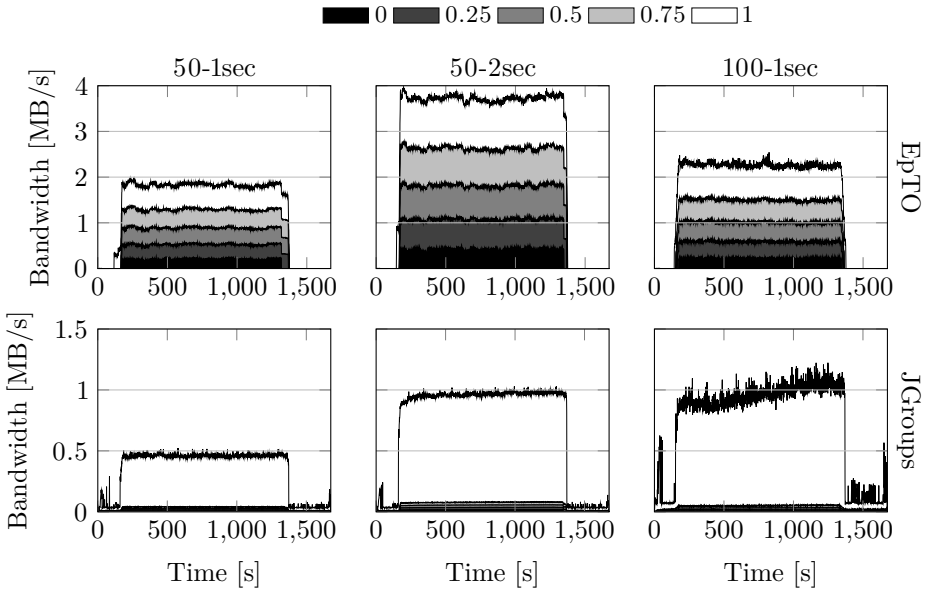

Figure 1: Throughput percentiles of a node during an experiment

uses a median bandwidth of approximately 1 MB/s for the smallest benchmark whereas JGroups uses a median bandwidth of less than 0.20 MB/s. However, in JGroups most of the work is done solely by the coordinator. We can clearly see this as the 100th percentile is much higher than the rest and uses approximately 0.50 MB/s.

Comparing EpTO and JGroups in terms of bandwidth when we increase the number of events sent per second, we can see the bandwidth doubling in both cases. In lower peers scenario such as the ones presented in Figure 1 JGroups has the upper hand. Since EpTO has a worse baseline we will reach the maximum bandwidth possible much quicker when increasing the event throughput.

Comparing EpTO and JGroups in terms of bandwidth when we increase the number of peers, EpTO is performing better than JGroups. Where JGroups basically has to double the bandwidth usage of the coordinator, EpTO only increases it by 50% or less. Jocelyn ▶*logarithmic I think as per the formula*◀. Thus in a scenario where we have many peers EpTO will be more efficient than JGroups at not reaching the maximum bandwidth.

In Figure 2 We analyze two different synthetic churns. In the first one we

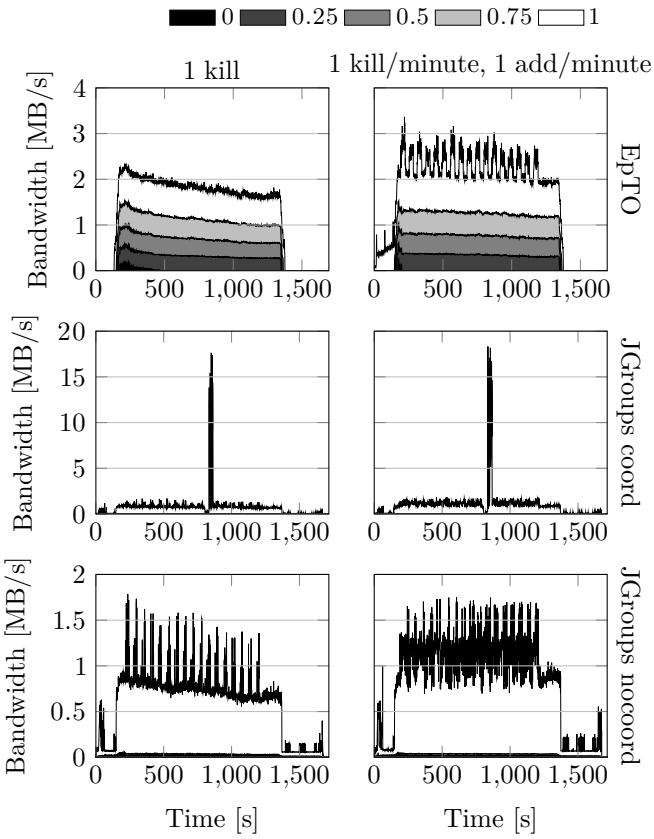

Figure 2: Throughput percentiles of a node during an experiment with churn

kill one node per minute. In the second one, we still kill one node per minute, but we immediately create a new one. For JGroups we ran the benchmarks once without killing the coordinator and once killing it.

All tests were run using 100 peers and a global throughput of 50 events per second (100-1sec). **Jocelyn** ►*I need to rephrase 50-1sec, 100-1sec etc. to be clearer*◄

We can see that the churn doesn't affect EpTO at all when there are only nodes leaving. We have small peaks when adding a node to 3 MB/s or less. Probably due to running the PSS initialization method on top of having one more node spreading rumors in the system. This is confirmed at the end of the plot where EpTO goes back to a normal Bandwidth after stabilization.

On the other hand, when killing the coordinator in JGroups we can see a huge spike in bandwidth, going from 1 MB/s to more than 15 MB/s. This is due to how JGroups operates when selecting a new coordinator.

Even when not killing the coordinator, JGroups suffers from the churn. We can see that each time the view changes, it generates an almost 100% increase in bandwidth usage. This is due to JGroups having to update the view and propagate it to every peer.

1.2 Total Bytes sent/received

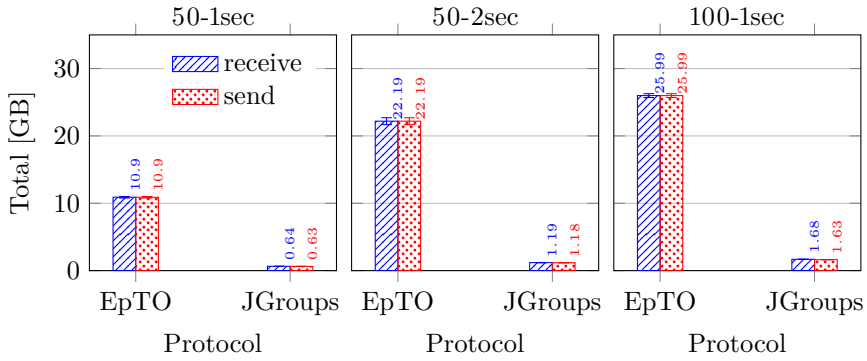

Figure 3: Total bytes sent/received during an average experiment

In Figure 3, EpTO has a worse baseline than JGroups. This is expected as EpTO sends $c*n*\log_2 n$ messages per events and JGroups sends at minimum n messages per event so we should have at least $c*\log_2 n$ more messages sent in EpTO if JGroups is perfect. Here we are well within this ratio.

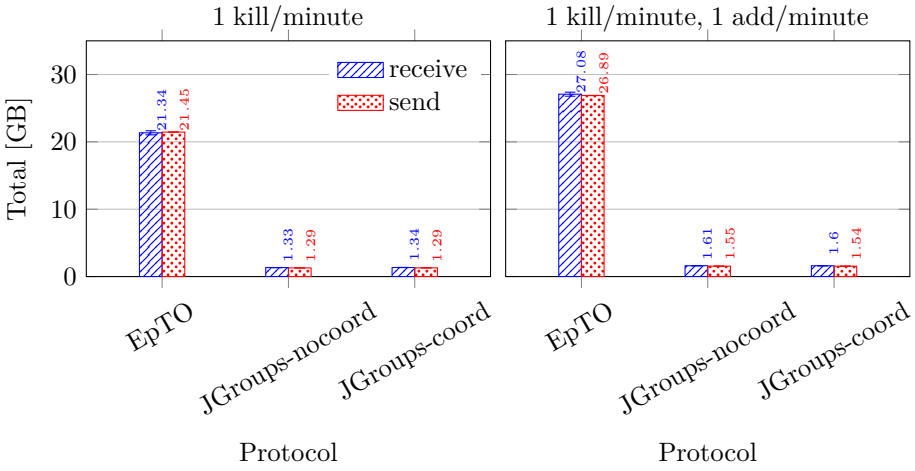

Figure 4: Total bytes sent/received during an average experiment with churn

In Figure 4 we can see that JGroups total bandwidth usage doesn't increase significantly compared to EpTO with churn. An explanation for this phenomenon is that JGroups takes more than a minute to find out the coordinator is effectively dead. During this time the bandwidth is practically not used. The big spike when the new coordinator is chosen compensates for this hole thus making the total bandwidth used appear as to not have changed.

The fact that killing the coordinator or not is of no effect on the total bandwidth used tends to show the hypothesis to be correct.

1.3 Local Times

Jocelyn ▶*Having a table for each percentile figure might be a good idea to put numbers for key percentiles (min,50th,max) for example*◀

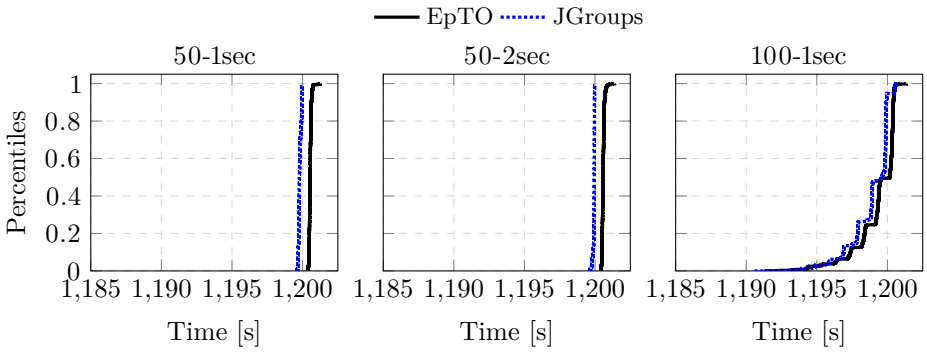

Figure 5: Local dissemination times

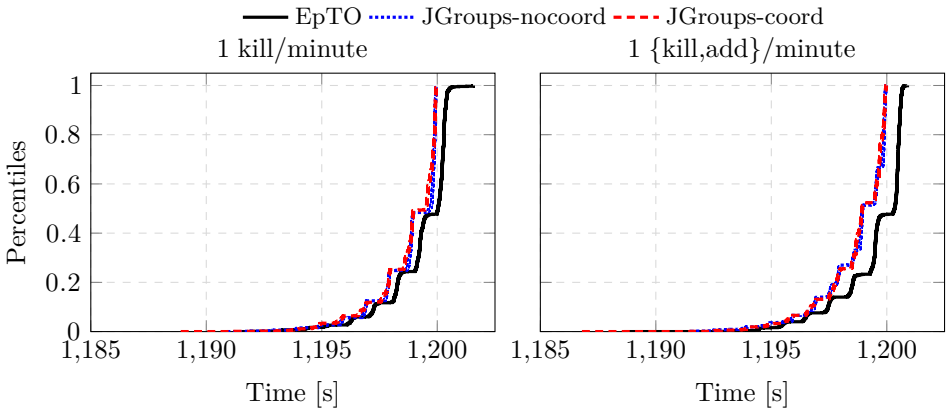

Figure 6: Local dissemination times with churn

In Figure 5, JGroups delivers all events quicker than EpTO in all scenarios, even when churn is involved as is shown in Figure 6. However, EpTO is not too far behind. The difference between EpTO and JGroups is likely to be even smaller when running them in a real WAN network due to the latency. EpTO in our configuration has a δ period of 100 ms and is thus handicapped against JGroups in a LAN environment, because it only increments the TTL of an event every 100 ms.

1.4 Global Times

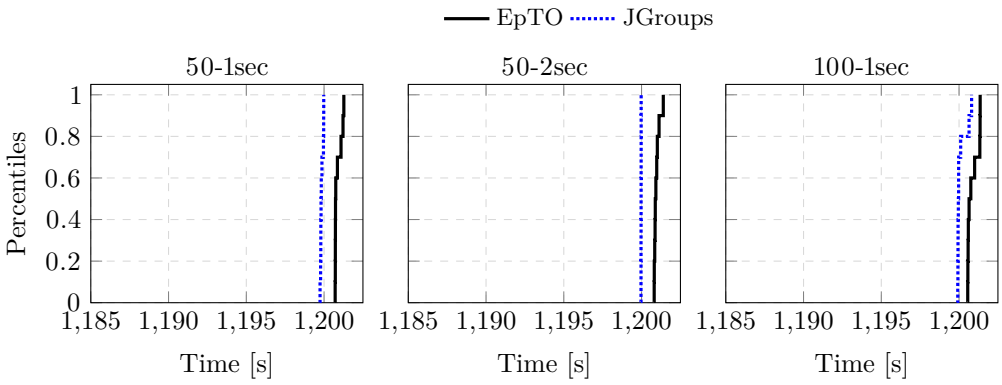

Figure 7: Global dissemination times

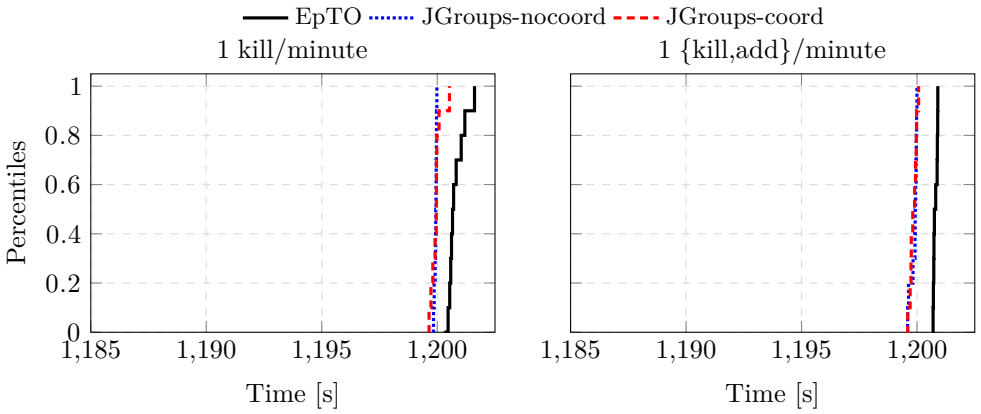

Figure 8: Global dissemination times with churn

We computed global times as well. They are represented in Figure 7 and Figure 8. These global times are of less interest than their local counterpart as they differences in clocks between hosts can skew this measurement.

Nonetheless, here too we can see that EpTO is consistently slower than JGroups for the same reason as stated in subsection 1.3.

1.5 Local Dissemination stretch

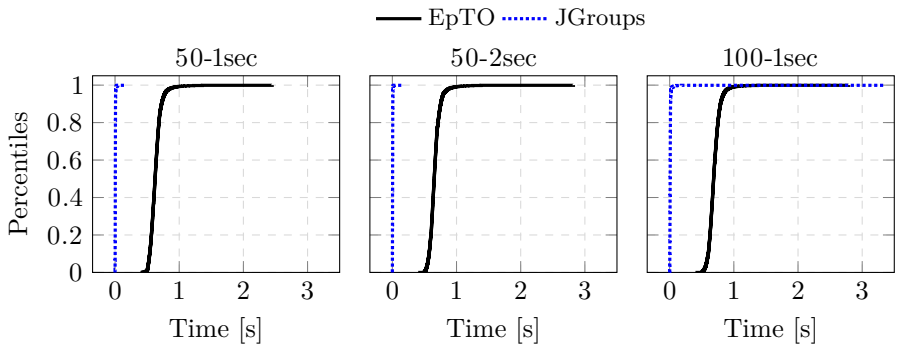

Figure 9: Local dissemination stretch

Jocelyn ▶*I wonder if we should split the plots (3 by 2) having a row for EpTO and a row for JGroups. This way we could show more for JGroups local dissemination stretch*◀In Figure 9, We can see the percentiles of the local dissemination stretch. The local dissemination stretch is the time measurement between the sending of an event by a peer and the delivery of this event locally.

JGroups is usually much faster than EpTO in a perfect environment. This is expected as the benchmarks involve a small number of nodes and are performed in a LAN environment with minmimal latency. The median dissemination stretch of JGroups is around 8 ms where as the median dissemiantion stretch of EpTO is around 685 ms. When increasing the number of peers, JGroups starts to have long delivery times

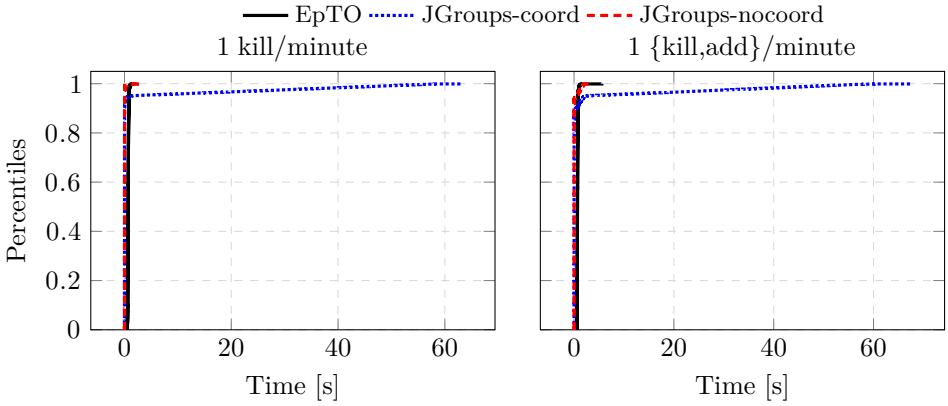

Figure 10: Local dissemination stretch with churn

In Figure 10 We can see a completely different picture. When under churn, the 90th percentile of JGroups is at 447 ms and the highest percentiles are at

more than 20 s with the highest dissemination stretch being 67.50 s. This effect is due to the coordinator dying as we clearly see that it does not happen when we do not kill it.

The median is bigger at around 11 ms, whether we kill the coordinator or not. This shows that there are some degradation in JGroups local dissemination stretch when under churn.

On the contrary, EpTO performs very well under churn. The median stays really close at 686 ms with the 99.9th percentile being at 1647 ms compared to 1366 ms when no churn is happening.

1.6 Events sent

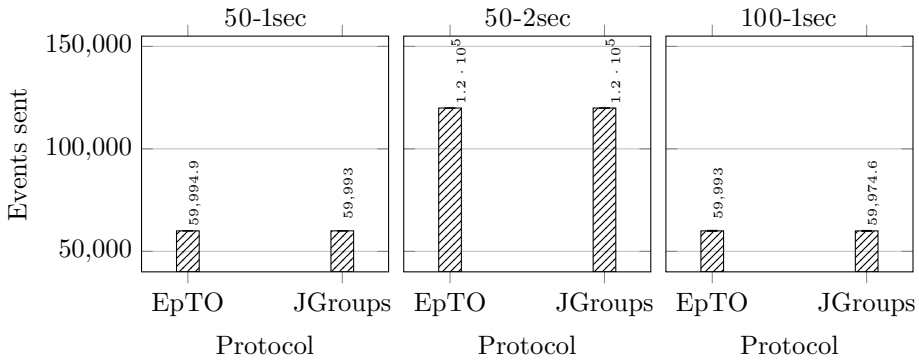

Figure 11: Total events sent per experiment on average

In Figure 11 we can see that both EpTO and JGroups deliver the same amount of events. This is expected in a perfect environment.

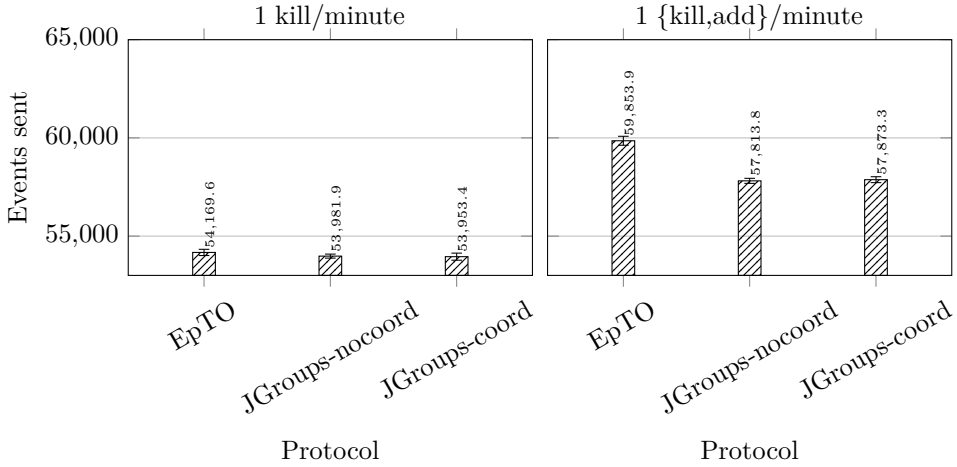

Figure 12: Total events sent per experiment during churn on average

In Figure 12 When only killing nodes, EpTO and JGroups again deliver the same amount of events.

However when killing and adding nodes, we notice that JGroups delivers more than 2000 less events than EpTO, even when the coordinator is not killed.

We are not sure why this is happening, but one hypothesis is that JGroups peers take a longer time to join the cluster and thus start sending events later. Since the benchmarks start and stop at a given time, they send less events in total.