1: Adapting Byzantine Agreement Algorithm in Synchronous Message Passing for Omission Faults with Polynomial Message Complexity

Analysis:
The problem of solving consensus when only omission faults can occurs is essentially guaranteeing that all processors exclude the same processors in case of an asymmetric omission error where some processors received a message but others didn't. In particular, this is solvable in a manner identical to the solution for byzantine agreement in a synchronous setting by simply ignoring the messages from processors that have exhibited omission errors. A processor can identify the processors exhibiting omission errors either if it itself doesn't receive a message from a particular processor in a certain round or if another processor informs it in its message that it didn't receive a message from said processor.

Algorithm:

Send your id to everyone ▷ Round 1
Let S = the set of all processor's ids (include your own) ▷ No message received represented by $\perp$
Send S to everyone ▷ Round 2
Let S_i = the set of ids received from P_i $\forall i$
while S and all S_is are not equal **do**
  for every processor P_j **do**
    if corresponding value in S or any of the sets $S_i = \perp$ or no message received from P_j in this round **then**
      Set value corresponding to P_j in S to $\perp$
  Send S to everyone ▷ Round k for some k $\leq$ t
Output majority of S

Running time:
Since the system has t faults there must be around in t + 1 rounds when no new faults occur (by pigeon hole principle) and in this round the termination condition will be satisfied. Therefore this algorithm will terminate in t + 1 rounds.

Correctness:
The algorithm is correct as since there is a round where no new faults occur there will be a point where all the received sets S_i s and S are equal. But then outputing the majority of S for every processor will have the same result as the local set S is identical across all processors. This ensures that the criteria of consensus will be met (triviality due to majority function and validity due to all having same majority). Moreover, this algorithm will have polynomial message complexity as in every round every processor sends a set containing n messages (assuming there is a message for itself as well) and all n processors do so in every round therefore, message complexity = $O(n^2)$ = poly(n)