1. One solution that could be implemented to begin cutting down on HOL blockage is the development of a secondary queueing scheme. Instead of having all traffic that comes from a specific link queueing at that link's entrance, if a packet is blocked for a number of cycles, it could be automatically sent to a secondary queue. When it reaches the top of that secondary queue, it will then have priority over other packets that need to go out on that link so that this secondary queue would remain as empty as possible. A major problem with this implementation is that in order to determine whether a certain link is blocked by a HOL problem, resources within the router would need to be assigned to keep track of datagrams, how long they've been around, and if the switching fabric is being used by another link when they need it. Additionally, this requires a transportation scheme to actually move the blocked packets around, something that could itself be blocked. Another possible way to combat HOL blockage is to give every inbound queue two different places to queue datagrams. During regular operation (meaning when the HOL is not blocked up) everything would function on one queue; however, when a HOL blockage is detected, all incoming traffic, as well as the traffic that was there already, would be split into both queues. Everything going to the same output link as the HOL blockage would be moved to this secondary queue through some means other than the switching fabric such as a bus. The primary problem with this solution is that it creates a secondary storage location that will be in operation far less often than desirable with an entirely separate set of storage available. Additionally, processing time would need to be dedicated to determining whether a certain link was being held up by HOL blockage or not, just like with the first proposed solution.
2.
    a. Host A would want to know if a message is fragmented so that future transmissions can be adjusted in size. If the MTU for a certain communication were smaller than expected, the fragmentation could cause an unnecessary amount of retransmission. For instance if all but 1 fragment of a fragmented datagram were to arrive at Host B, the entire datagram would need to be resent, and if Host A weren't away of it being previously fragmented, then it would need to be fragmented again, which could lead to further data loss, perpetuating this cycle.
    b. First, a new IP header for the first datagram is generated. This 20 byte header includes all of the relevant fragmentation data such as its ID and the proper routing information. Next, the protocol header from the datagram is placed into the new fragmented datagram. From there, the datagram is filled up to whatever the new MTU size will be. Each subsequent fragment will be created by first putting down the IP fragment header and then filling the rest of the fragment with data from the unfragmented datagram. Once all of the date from the unfragmented datagram has been placed into one of the fragments, the transmission has been properly split up for a new MTU.
    c. Once fragmented and forwarded:

i. If one fragment is lost, the complete datagram is unable to be reconstructed. Thus the entire packet will be resent (in the case of TCP) or discarded (in the case of UDP).
ii. The fragments will be fragmented in the same process as is described above in part (b). Then when reassembled, they will indicate that they are a part of a larger fragment through their protocol header.
iii. Once a fragmented datagram arrives at its destination, and is recognized as a fragmented datagram, it is reconstructed. First, the destination will ensure that all of the fragments have arrived. If not, then the fragments will either discarded or the destination will request the data be re-transmitted. Assuming everything has arrived properly, the first fragment is identified by checking for an offset of 0. Then, the fragment's headers are stripped off, and the data is reassembled using the offset to determine the order. Once all of the fragments have been reassembled, the datagram can be processed as if it had arrived in a single piece.

3.

a. 4 fragments are generated to send the full datagram onwards.
b. The below table demonstrates the relevant pieces of the generated IP header for a fragmented datagram.

Fragment #	Size (bytes)	ID	Offset (bytes)	Flag
Number 1	576	422	0	1
Number 2	576	422	72	1
Number 3	576	422	144	1
Number 4	128	422	216	0