Protocol Assumptions

1. No node failures.
2. No message loss.
3. No message corruption.

Key Mechanisms

1. Timestamps: Each message includes a sender-assigned timestamp, ensuring consistent ordering within the same process.
2. Acknowledgments: Each process acknowledges every received message, i.e whenever a new message added to the local queue.
3. Queue Ordering: The message queue is sorted based on timestamps and sender IDs, enforcing a delivery order.
4. Delivery Condition: Messages are only delivered after receiving acknowledgments from all processes.

Proof by Contradiction

Assume:
Two processes, A and B, deliver messages M and N with timestamps i:M and j:N, respectively, in different orders
(without loss of generality, i < j).

B's delivery of j:N implies:
B has received all acknowledgments for N, including the acknowledgment from A.

Contradiction:
Case 1: B receives some acknowledgments for N before i:M

1. **Assume:** B receives acknowledgments from some processes (excluding A) for message N before receiving message i:M.
2. **B still needs A's acknowledgment for N:** B cannot deliver N yet because the delivery condition requires all acknowledgments.
3. **A delivers i:M:** A delivers i:M after receiving all acknowledgments for it, including B's acknowledgment for i:M (due to the delivery condition).
4. **A sends acknowledgment for N:** Upon receiving N, A sends its acknowledgment.
5. **B receives A's acknowledgment for N:** This completes all required acknowledgments for N at B.

6. **Contradiction:** Based on the timeline, B received i:M before receiving the final acknowledgment (A's) for j:N. Since A's message has a smaller timestamp (i < j), Queue ordering brings A's message to front of queue. However, B is assumed to have delivered j:N before i:M, contradicting the total order property.

Case 2: B receives all acknowledgments for N (including A's) before i:M

1. **Assume:** B receives all acknowledgments for N, including A's, before receiving message i:M.
2. **B satisfies delivery condition for N:** B can now deliver N based on having all acknowledgments.
3. **However, A cannot have delivered i:M yet:** A's delivery requires receiving all acknowledgments for i:M, including B's acknowledgment. Since B hasn't received i:M yet, it cannot have acknowledged it.
4. **Contradiction:** This scenario leads to a deadlock-like situation, where B waits to deliver N until it receives i:M, but A waits to deliver i:M until it receives B's acknowledgment for i:M which B cannot send without receiving i:M itself. This contradicts the assumption of both processes successfully delivering their messages.

Conclusion:
Since the initial assumption leads to a contradiction, all processes must deliver messages in the same total order based on the combined effect of message IDs, timestamps, and the acknowledgment scheme.

The contradiction arises from the combined effect of:

1. **FIFO channel:** Guaranteeing messages are delivered in the order they are sent.
2. **Timestamp-based ordering:** Ensuring consistent ordering within a process and breaking ties between messages from different processes.
3. **Acknowledgment scheme:** Ensuring all processes have received a message before any process delivers it.