Solution 2.4

The **Comparison** of different channels are as follow.

	Reliable Channel	Semi-Reliable Channel : **Deterministic**	Semi-Reliable Channel : **Random**
No Duplication	No message **delivered** more than once	A message is **received** at most once	A message is received at most once
No Creation	No message is **delivered** unless it was **broadcast**	No message is **received** unless some process did **send** it	No message is received unless some process did send it
Validity	For Correct P_i & P_j then for every message broadcast by P_i is **eventually delivered** by P_j	For Correct P_i & P_j , It provides **50% assurance** for every message to be received by P_j. Anyhow One of two m would be received.	For Correct P_i & P_j , It provides **25% assurance** for every message to be received by P_j. Anyhow One of two m would be received.
Agreement	If correct P_i delivers m then P_j eventually **delivers**	It cann't provide any delivery assurance as there is no agreement rule defined.	**No delivery assurance**

Both of them don't fulfill the criteria for a Reliable channel.

Statement

Reliable channel is stronger and stricter than Semi-Reliable Channel because

1. Semi-Reliable channel can be derived from Reliable channel.
2. Reliable channel is stronger than Semi-Reliable channel not vice-versa.

Proof

Algorithm

1. Process P sends every second message m (or drops one of the two messages) to every other process including itself.
2. Every process which receives m for the first two time (one of two) sends it to every other process (except the sender) and delivers it.

$\forall$ message m $\in \{m_1, m_2 \ldots\ldots\ldots\ldots m_n\}$

(m_1, m_2) Process P_i (m_1 || m_2) ----------> P_j (m_1 || m_2)

P_i drops one of two message deterministically or random as given and similarly P_j

The above algorithm satisfies the assumption that **Semi-Reliable channel could be derived from** more stricter form of **Reliable channel**.

Reliable channel could not be derived from Semi-Reliable channel as nearly **half of the messages are being dropped** from being eventually received by P_j and hence preventing eventual delivery of every messages by correct processes P_i and P_j .