# 1. Zielstellung

**gegeben:**

Kommunikationssystem aus vielen einzelnen Prozessoren, z.T. mit ***schwerwiegenden Fehlern*** („byzantine“)

**bereits bekannt:**

***Algorithmen*** zur Lösung bestimmter Probleme bei ***leichten Fehlern*** („crash“)

**erwünscht:**

***Weiterverwendung*** der bereits bekannten Algorithmen auch bei Systemen mit schwerwiegenden Fehlern

**also gesucht:**

***Protokollschicht*** zwischen Kommunikationssystem und Algorithmus, die dem Algorithmus ein System mit leichten Fehlern vortäuscht

## Modell des zugrundeliegenden Kommunikationsystems

- Topologie: Graph vollständig verbunden
- Kommunikation ausschließlich über „broadcast“
- Einteilung der n Prozessoren: (n-f) nonfaulty, f faulty processors

## zugesicherte Eigenschaften

**„Nonfaulty Integrity“**: „If nonfaulty processor pi receives (m,k) from nonfaulty processor pj then pj sends m in round k.“

**„Nonfaulty Liveness“**: „If nonfaulty processor pi sends m in round k, then nonfaulty processor pj receives m in round k.“

# 3.Verbesserung der Fehlertoleranz (synchroner Fall)

DREI Unterschiede zwischen „byzantine processors“ (bp) und „crash processors“ (cp):

- bp können beim broadcast ***unterschiedliche Nachrichten*** an verschiedene Prozessoren verschicken

- ***Inhalt*** einer Nachricht von einem bp kann falsch sein

- Fehler können ***in jeder Runde wieder*** auftreten

# „identical byzantine failure model“

**„Nonfaulty Integrity“**: If nonfaulty processor pi receives (m,k) from nonfaulty processor $p_j$ then $p_j$ sends m in round k.

**„Faulty Integrity“**: If nonfaulty processor $p_i$ receives (m,k) from $p_h$ and nonfaulty processor $p_j$ receives (m´,k) from $p_h$, then m=m´.

**„No Duplicates“**: Nonfaulty processor $p_i$ receives only one message with tag k from $p_j$

**„Nonfaulty Liveness“**: If nonfaulty processor $p_i$ sends m in round k, then nonfaulty processor $p_j$ receives (m,k) in round k.

**„Faulty Liveness“**: If nonfaulty processor $p_i$ receives (m,k) from processor $p_h$ in round r, then nonfaulty processor $p_j$ receives (m,k) from ph by round r+1.

Algorithmus „identical byzantine“ (Teil 1)

Initially $S = \emptyset$ and *accepted* $= \emptyset$

round $(k, 1)$: in response to id-send$_i(m)$:

1: Byz-send$_i(S \cup \{\langle init,m,k\rangle\})$

2: Byz-recv$_i(R)$

3: $S := \{ \langle echo,m',k,j\rangle$ : there is a single $\langle init,m',k\rangle$ in $R$ with sender $p_j$ $\}$

4: $S := S \cup \{ \langle echo,m',k',j\rangle : k' < k$ and
$m'$ is the only message for which at least $n - 2f$
$\langle echo,m',k',j\rangle$ messages have been received in this round $\}$

5: *accepted* $:= \{(m', k')$ with sender $p_j$ : at least $n - f$ $\langle echo,m',k',j\rangle$
messages have been received in this round and first-accept$(k', j)\}$

Algorithmus „identical byzantine“ (Teil 2)

round $(k, 2)$:

6: Byz-send$_i$ $(S)$

7: Byz-recv$_i$ $(R)$

8: $S := \{ \langle echo, m', k', j \rangle : k' \leq k$ and
$m'$ is the only message for which at least $n - 2f$
$\langle echo, m', k', j \rangle$ messages have been received in this round $\}$

9: $accepted := accepted \cup \{(m', k')$ with sender $p_j$ : at least $n - f$
$\langle echo, m', k', j \rangle$ messages have been received in this round
and first-accept$(k', j)\}$

10: id-recv$_i$(*accepted*)

# „omission model“

wie byzantine model, aber:

**„Integrity“**: Every message received by processor $p_i$ from processor $p_j$ in round k was sent in round k by $p_j$.

**„Nonfaulty Liveness“**: The message sent in round k by nonfaulty processor $p_i$ is received by nonfaulty processor $p_j$ in round k.

# „crash failure model“

wie omission model, aber:

**„Integrity“**: Every message received by processor pi from processor $p_j$ in round k was sent in round k by $p_j$.

**„Nonfaulty Liveness“:** The message sent in round k by nonfaulty processor $p_i$ is received by processor $p_j$ in round k.

**„Faulty liveness“:** If processor $p_i$ fails to receive processor $p_j$´s round k message, then no processor receives any message from $p_j$ in round k+1.