

Coding Approaches to Fault Tolerance in Dynamic Systems

Christoforos Hadjicostis

Digital Signal Processing Group
Massachusetts Institute of Technology

FAULT-TOLERANT SYSTEMS

Fault tolerance describes ability to

- Withstand internal failures
- Produce desirable overall “behavior”

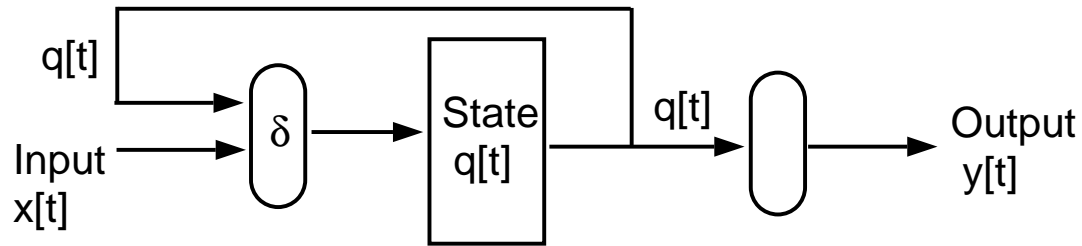
Necessary or desirable in

- Life-threatening circumstances (military, transportation, medical)
- Systems in inaccessible environments (space missions)
- Reliable systems from unreliable components (faster, less expensive)

WORK ON FAULT TOLERANCE

- Communication systems (channel noise, error-correcting codes)
- Computational circuits (hardware failures, modular redundancy)
 - Each component fails with *constant* probability
 - Earlier work by von Neumann, Shannon, Winograd, Elias and others.
Later work by Gács, Pippenger, Feder, Reischuk.
- Special-purpose systems (Algorithm Based Fault Tolerance)
 - Protect against *fixed* number of failures
 - Recent work by Abraham et al., Redinbo, Musicus, Beckmann.

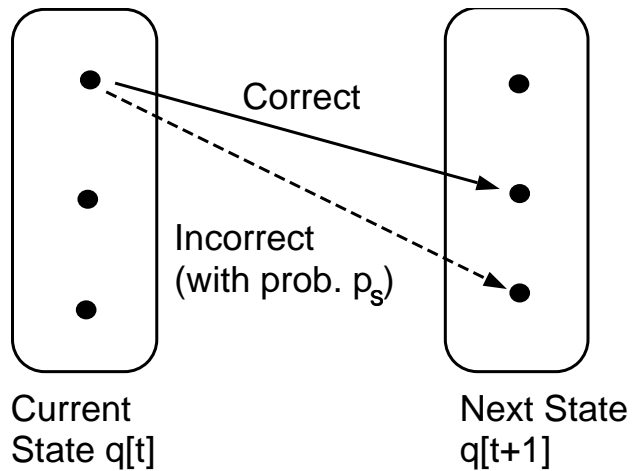
DYNAMIC SYSTEMS



State Evolution of system S : $q[t + 1] = \delta(q[t], x[t])$

Examples: digital filters, encoders/decoders, computer simulations

Failures in state transitions:

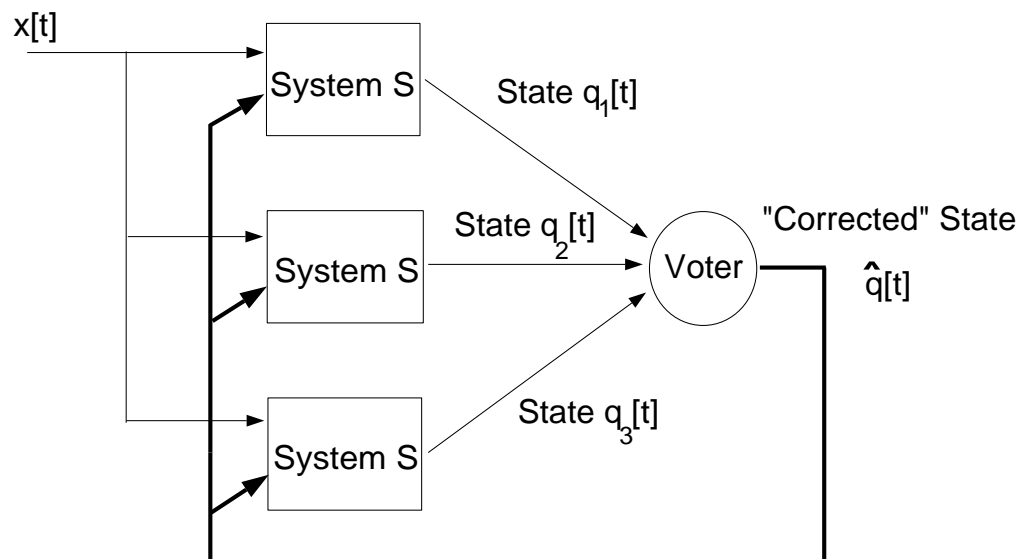


After L time steps:

$$\Pr[\text{correct state trajectory}] = (1 - p_s)^L$$

Number of steps is costly

TRADITIONAL APPROACH: MODULAR REDUNDANCY (VON NEUMANN, 1956)



Problems:

- Replication
- If voter fails with prob. p_v , then after L steps:

$$\Pr[\text{correct state trajectory}] \leq (1 - p_v)^L$$

THESIS OUTLINE

- **Chapters 2 through 4:**

Avoiding replication via general system embeddings:

- Machines (group/semigroup machines)
Algebraic approach, no connections to hardware
- Linear time-invariant (LTI) dynamic systems
- Linear finite state machines (LFSM's)

- **Chapter 5:**

Petri net embeddings for monitoring failures in discrete event systems

- **Chapter 6:**

Handling failures in the error correcting mechanism

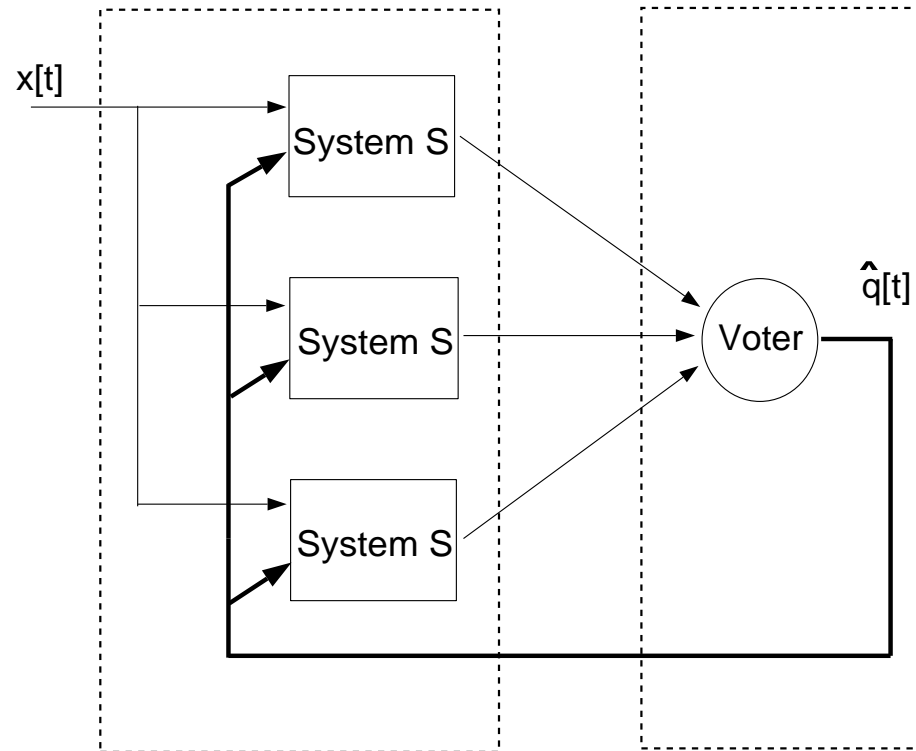
TALK OUTLINE

- Fault-tolerant dynamic systems
- **Avoiding replication**
 - General approach
 - Examples from linear dynamic systems
- Failures in error correcting mechanism
 - Distributed voting schemes
 - Reliable systems with *constant* redundancy
- Future research

AVOIDING REPLICATION

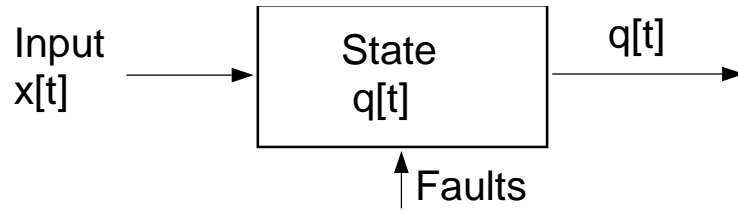
Redundant
Implementation

Error
Correction

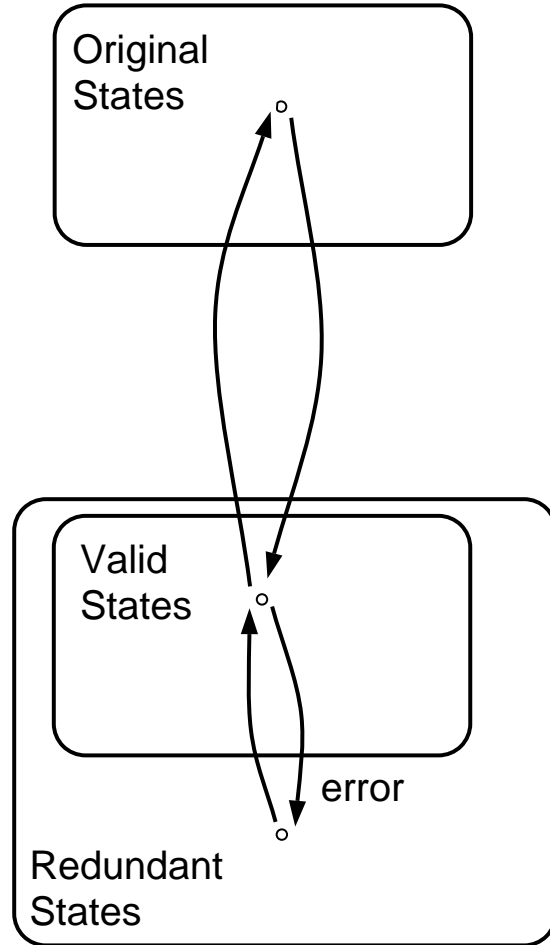
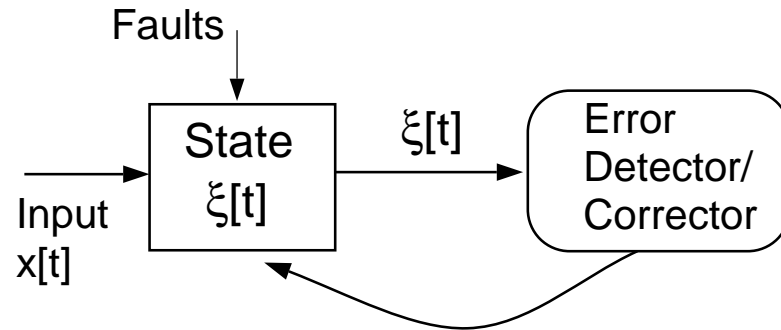


Is replication necessary?

REDUNDANT IMPLEMENTATIONS



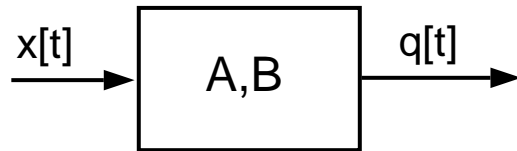
Replace with larger dynamic system:



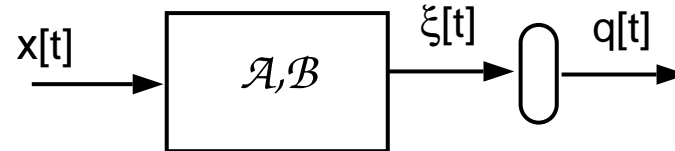
OUTLINE

- Fault-tolerant dynamic systems
- Avoiding replication
 - General approach
 - **Examples from linear dynamic systems**
- Failures in error correcting mechanism
 - Distributed voting schemes
 - Reliable systems with *constant* redundancy
- Future research

REDUNDANT LTI DYNAMIC SYSTEMS



$$\mathbf{q}[t + 1] = \mathbf{A}\mathbf{q}[t] + \mathbf{B}\mathbf{x}[t]$$



$$\xi[t + 1] = \mathcal{A}\xi[t] + \mathcal{B}\mathbf{x}[t]$$

- Concurrent Simulation: $\mathbf{q}[t] = \mathbf{L}\xi[t]$
- State Constraints: $\xi[t] = \mathbf{G}\mathbf{q}[t]$
- Fault Detection: If ξ is *not* in column space of \mathbf{G} , or

$$\mathbf{P}\xi[\cdot] \neq \mathbf{0}$$

where $\mathbf{P}\mathbf{G} = \mathbf{0}$

CHARACTERIZATION OF REDUNDANT LTI DYNAMIC SYSTEMS

$$\left. \begin{array}{l} \text{Original System} \\ \mathbf{q}[t+1] = \mathbf{A}\mathbf{q}[t] + \mathbf{B}\mathbf{x}[t] \end{array} \right\} \begin{array}{l} \xi[\cdot] \xrightarrow{=\mathbf{G}\mathbf{q}[\cdot]} \\ \mathbf{q}[\cdot] \xleftarrow{=\mathbf{L}\xi[\cdot]} \end{array} \left\{ \begin{array}{l} \text{Redundant Implementation} \\ \xi[t+1] = \mathcal{A}\xi[t] + \mathcal{B}\mathbf{x}[t] \end{array} \right.$$

Standard Redundant Implementations:

$(\mathcal{A}, \mathcal{B})$ is a redundant implementation for (\mathbf{A}, \mathbf{B}) iff $(\mathcal{A}, \mathcal{B})$ is similar to the following standard form:

$$\xi_\sigma[t+1] = \underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix}}_{\mathcal{A}_\sigma} \xi_\sigma[t] + \underbrace{\begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix}}_{\mathcal{B}_\sigma} \mathbf{x}[t]$$

for some matrices $\mathbf{A}_{12}, \mathbf{A}_{22}$

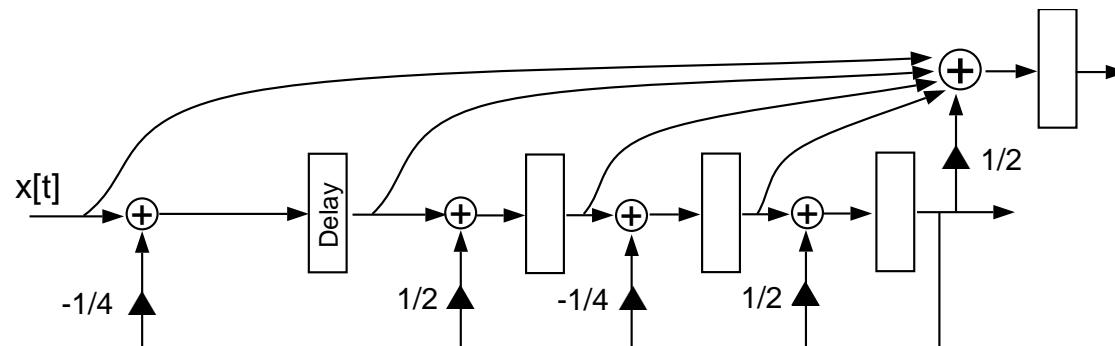
Specifically: Invertible \mathcal{T} such that

$$\mathcal{A}_\sigma = \mathcal{T}^{-1} \mathcal{A} \mathcal{T}$$

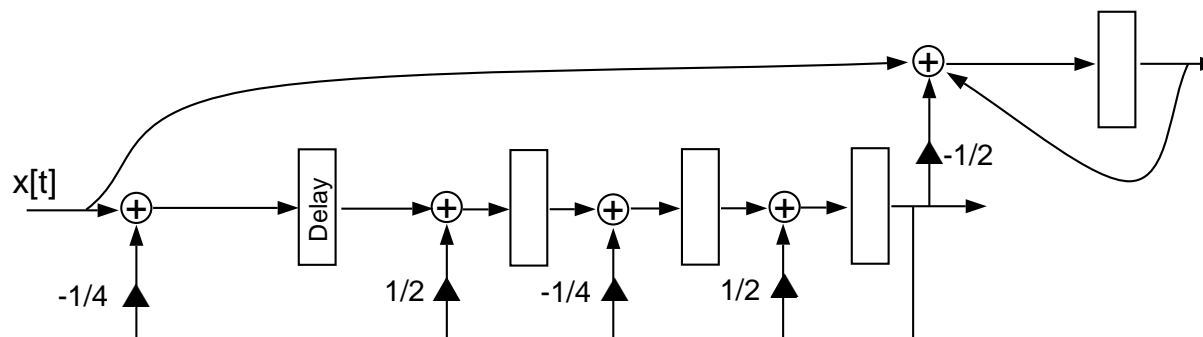
Related work: Šiljak's "inclusion principle".

DIFFERENT REDUNDANT IMPLEMENTATIONS FOR CHECKSUM SCHEME

Traditionally (Chatterjee and d'Abreu, Abraham et al., Reed):



Using previous theorem:



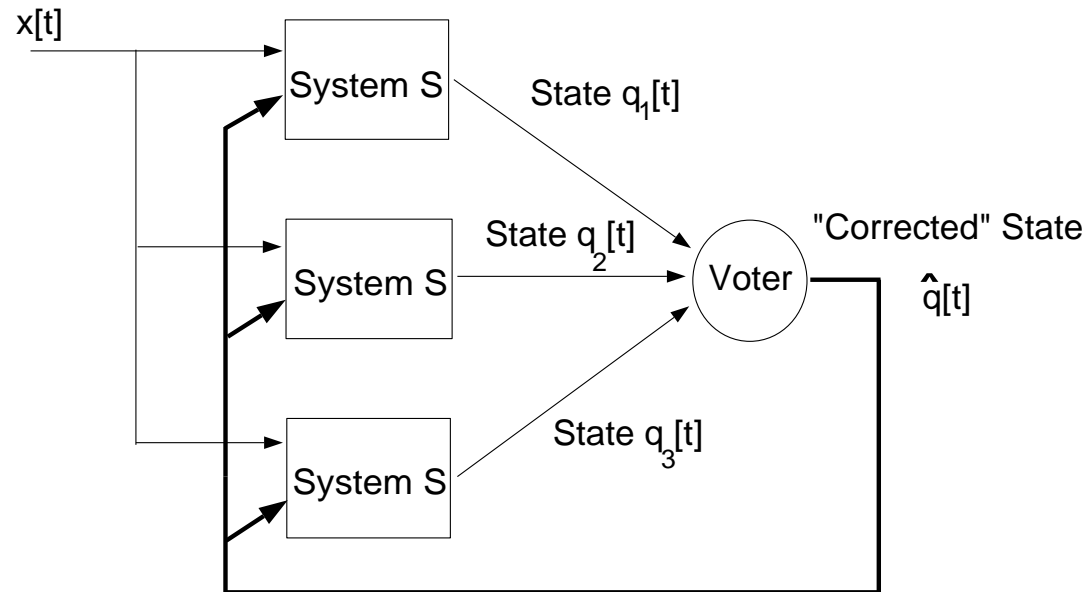
RELATED EXTENSIONS

- Immediate implications for linear systems:
 1. Reduced hardware
 2. Periodic checking for transient failures
 3. Systematic reconfiguration for permanent failures
- Extensions to other “linear” systems (LFSM’s, max-plus systems)
- Algebraic framework:
 - Appropriate mappings and redundant implementations
 - Algebraic error correction
- Petri net models of discrete event systems (DES)
 - Monitoring schemes for DES, different tradeoffs
 - Potential applications: manufacturing systems, networked systems, protocols

OUTLINE

- Fault-tolerant dynamic systems
- Avoiding replication
 - General approach
 - Examples from linear dynamic systems
- **Failures in error correcting mechanism**
 - **Distributed voting schemes**
 - Reliable systems with *constant* redundancy
- Future research

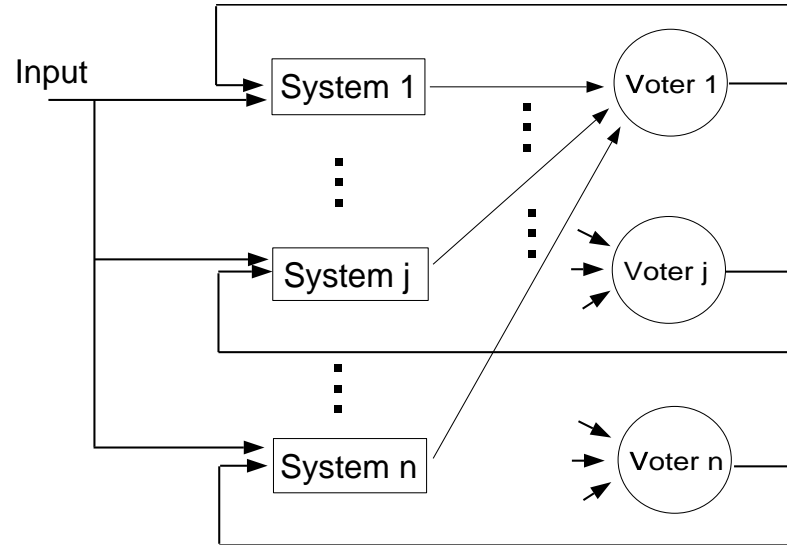
MODULAR REDUNDANCY



If voter fails with prob. p_v , then after L steps:

$$\Pr[\text{correct state trajectory}] \leq (1 - p_v)^L$$

DISTRIBUTED VOTING SCHEMES



At each step:

- Systems fail with prob. p_s
- Voters fail with prob. p_v

$$\Pr[\text{overall failure at or before time } L] \leq L \sum_{i=n/2}^n \binom{n}{i} p^i (1-p)^{n-i} \quad p \equiv p_v + (1-p_v)p_s$$

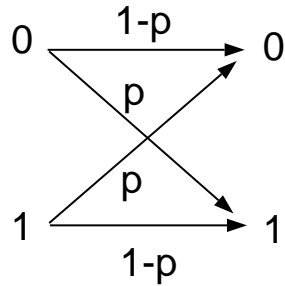
Result: Probability decreases exponentially with n if $p < 1/2$

Related: Compressor graphs (Margulis, Pippenger), stable memories (Taylor, Kuznetsov)

OUTLINE

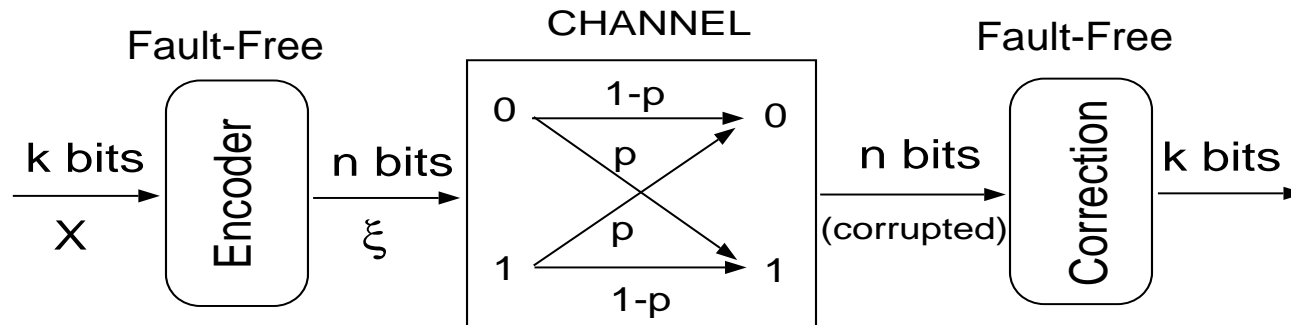
- Fault-tolerant dynamic systems
- Avoiding replication
 - General approach
 - Examples from linear dynamic systems
- Failures in error correcting mechanism
 - Distributed voting schemes
 - **Reliable systems with *constant* redundancy**
- Future research

ANALOGY WITH UNRELIABLE COMMUNICATION LINK



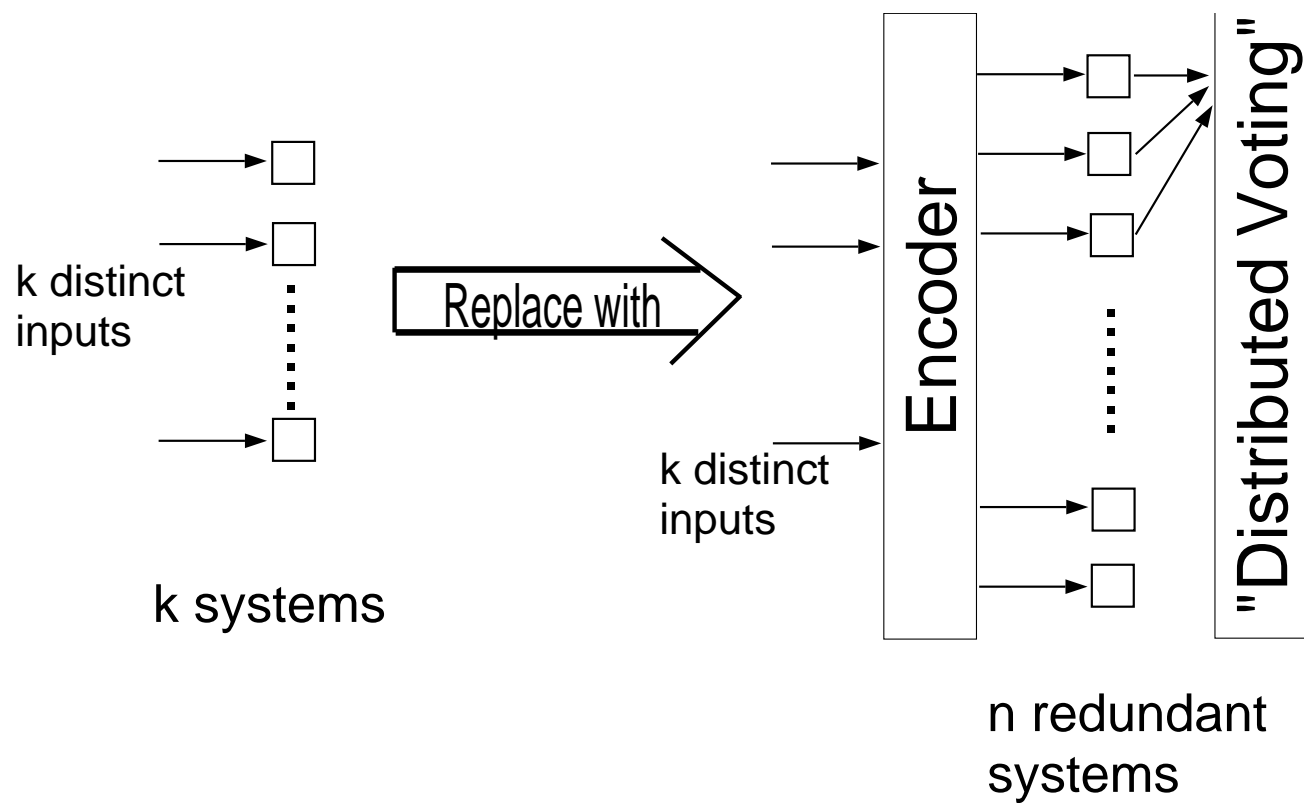
Send same bit n times:

$$\Pr[\text{error}] = \sum_{i=n/2}^n \binom{n}{i} p^i (1-p)^{n-i}$$



Shannon: Can send k bits with arbitrarily small error by sending n bits as long as $\frac{k}{n} < C$ (where C is the *channel capacity* and depends only on p)

CAN WE EMBED k DISTINCT SYSTEMS INTO n REDUNDANT SYSTEMS?



EMBEDDING k DISTINCT SYSTEMS INTO n REDUNDANT SYSTEMS

Consider an LFSM:

$$\mathbf{q}[t + 1] = \mathbf{A}\mathbf{q}[t] \oplus \mathbf{b}x[t]$$

Examples: encoders/decoders, counters, sequence generators

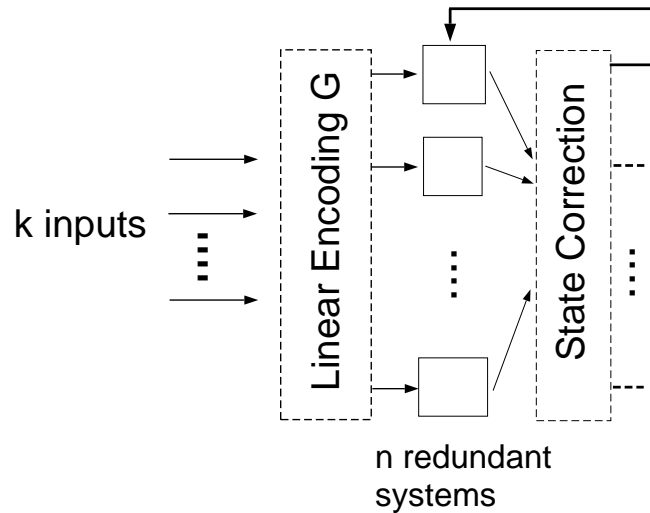
Embed k instantiations into n systems using (n, k) linear code:

$$\left[\xi_1[\tau] \ \xi_2[\tau] \ \cdots \ \xi_n[\tau] \right] \equiv \left[\mathbf{q}_1[\tau] \ \mathbf{q}_2[\tau] \ \cdots \ \mathbf{q}_k[\tau] \right] \mathbf{G}^T$$

if

$$\left[\xi_1[t + 1] \ \xi_2[t + 1] \ \cdots \ \xi_n[t + 1] \right] = \mathbf{A} \left[\xi_1[t] \ \xi_2[t] \ \cdots \ \xi_n[t] \right] \oplus \underbrace{\mathbf{b} \left(\left[x_1[t] \ x_2[t] \ \cdots \ x_k[t] \right] \mathbf{G}^T \right)}_{e \left(\left[x_1[t] \ x_2[t] \ \cdots \ x_k[t] \right] \right)}$$

PROTECTION OF k SYSTEMS USING n REDUNDANT SYSTEMS



If fault-free state correction, treat it like a communication problem:

- Shannon's approach achieves arbitrarily low probability of failure *given* no failure in previous step

$$\Pr[\text{error per step}] = \Pr[\text{failure at } \tau \mid \text{no failure at } \tau - 1]$$

- $\Pr[\text{failure at or before time } L] < L \Pr[\text{error per step}]$

PROTECTION OF k DISTINCT SYSTEMS USING n REDUNDANT SYSTEMS

Theorem:

Using **constant redundancy** per system, we can embed k distinct LFSM's into n redundant LFSM's such that

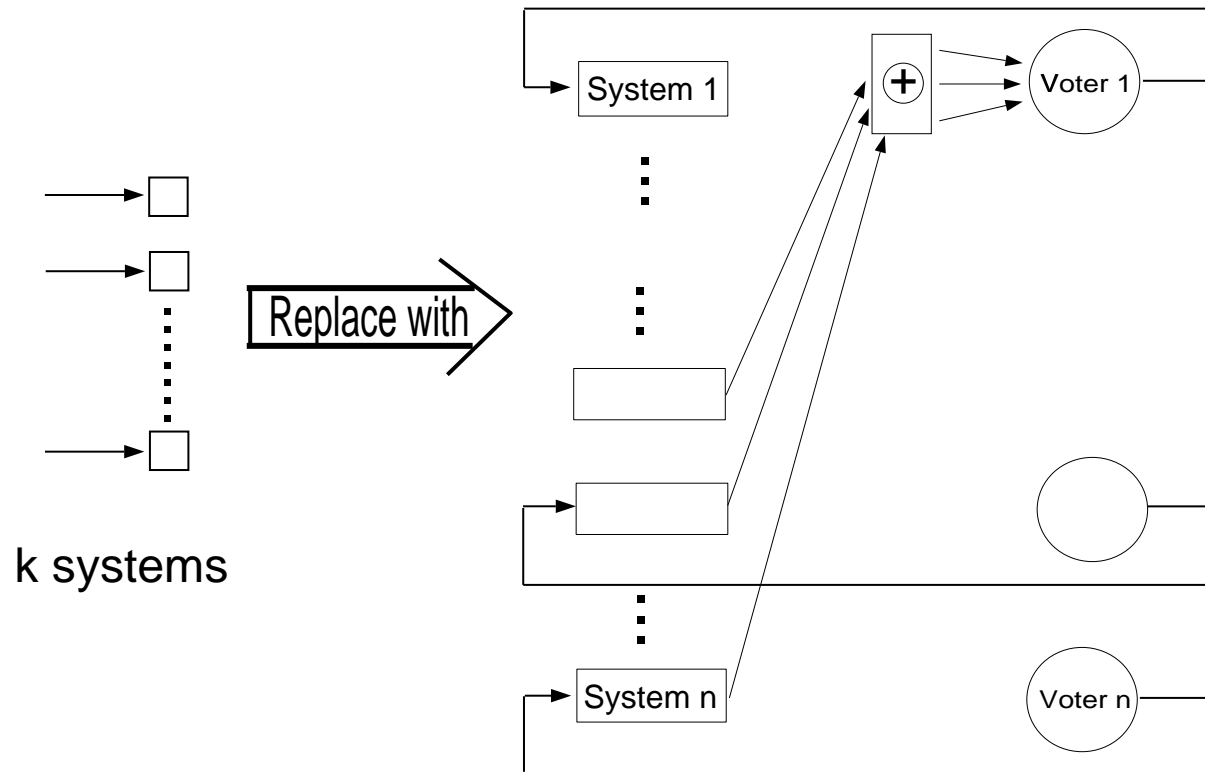
$$\Pr[\text{failure at or before time } L] < L C k^{-\beta}$$

Proof:

- LFSM's use unreliable XOR gates (2-input)
- Encoding uses low density parity check (LDPC) codes (Gallager 1963)
- Decoding done using XOR gates and voters
(techniques studied by Gallager (1963) and Taylor (1968))

Related work: Spielman, Gács

HOW IS THIS ACHIEVABLE?



- Each voter uses inputs from constant number of systems
- Critical cycle increases logarithmically with n
- Bound probability of *error propagation*

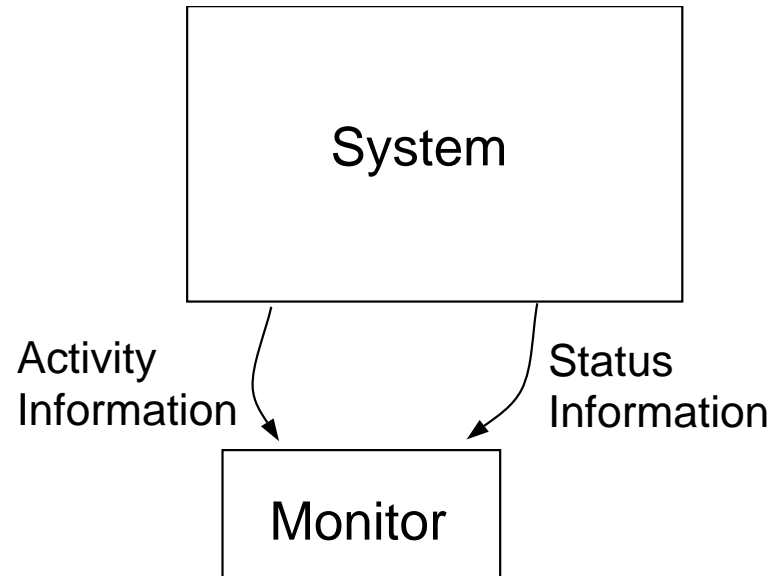
FUTURE RESEARCH DIRECTIONS

- Practical implementations:
 - Fast and inexpensive encoders/decoders
 - Digital signal processing applications (linear filters)
- Generalizations (finite state machines, computer simulations)
 - Reliable systems out of unreliable, fast, inexpensive components
- Extensions to permanent failures (reconfiguration)
- Theoretical work:
 - Bounds, “computational capacity”
 - Low-complexity error correction
(iterative schemes, sequential decoding, “turbo” codes etc.)

SUMMARY OF CONTRIBUTIONS

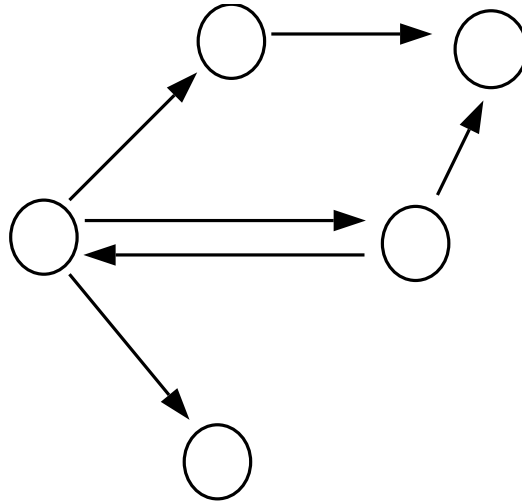
- Redundant implementations for dynamic systems
 - Exploited redundant system structure and dynamics
 - Generalized/combined modular redundancy and checksum schemes
 - Examples: LFSM's, LTI dynamic systems, Petri nets, algebraic systems
- Allowed failures in error correcting mechanism
 - Distributed voting schemes
 - Efficient constructions of reliable parallel LFSM's

SYSTEM MONITORING



- Systematic/simple construction of monitoring schemes
- Robustness to incomplete or incorrect information
- Hierarchy, distributivity
- Potential applications (manufacturing systems, system testing/verification, power systems)

PROCESSING/COMMUNICATION NETWORKS



- Communicating/interacting processors
- Fast decoding and routing
- Network reconfiguration under failures

OTHER INTERESTS

- Fault-tolerant algorithms
- Software reliability
- Probabilistic encoding
- Real-time systems

OTHER INTERESTS

Practical (apply error detection/correction techniques):

- System monitoring/testing/verification
- Software reliability
- Real-time systems

Theoretical:

- **Networking/Communication**
 - Communicating/interacting processors
 - Probabilistic encoding, probabilistic routing
 - Network reconfiguration, performance guarantees
- **Computational Architectures**
 - Quantum computation