Software-based Failure Detection in Programmable Network Interfaces

Yizheng Zhou, Vijay Lakamraju, Israel Koren, C.M. Krishna

Architecture and Real-Time Systems (ARTS) Lab
Department of Electrical & Computer Engineering
University of Massachusetts at Amherst

ARTS

Introduction

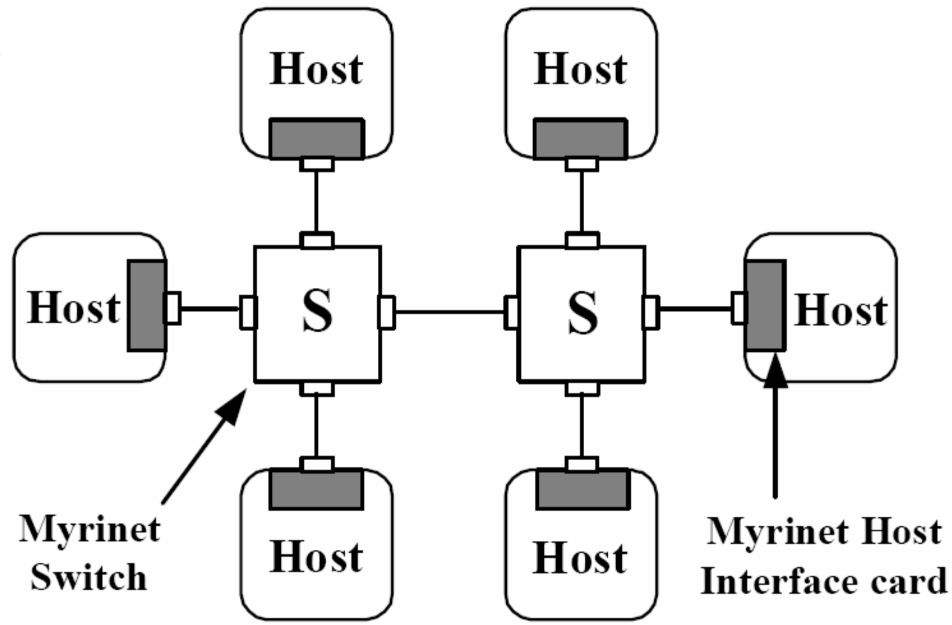
- Complex network interfaces
  - Typical Ethernet controller: 10 thousand gates
  - IXP1200: 5 million gates
- Transient faults: a major reliability concern
  - Neutrons from cosmic rays
  - Alpha particles from packaging material
- Software-based fault tolerance approaches
  - Pros: Less expensive than
    - Custom hardware
    - Massive hardware redundancy
  - Cons: Overhead
    - Performance degradation
    - Increased code size

Software-Based Failure Detection

- Network interface failures
  - Hardware failures
  - Software failures
    - The instruction and data of the Network Control Program (NCP) in the local memory.
- Requirements for failure detection of network interfaces
  - Limited performance impact
    - Performance is critical for high-speed network interface
  - Good failure coverage

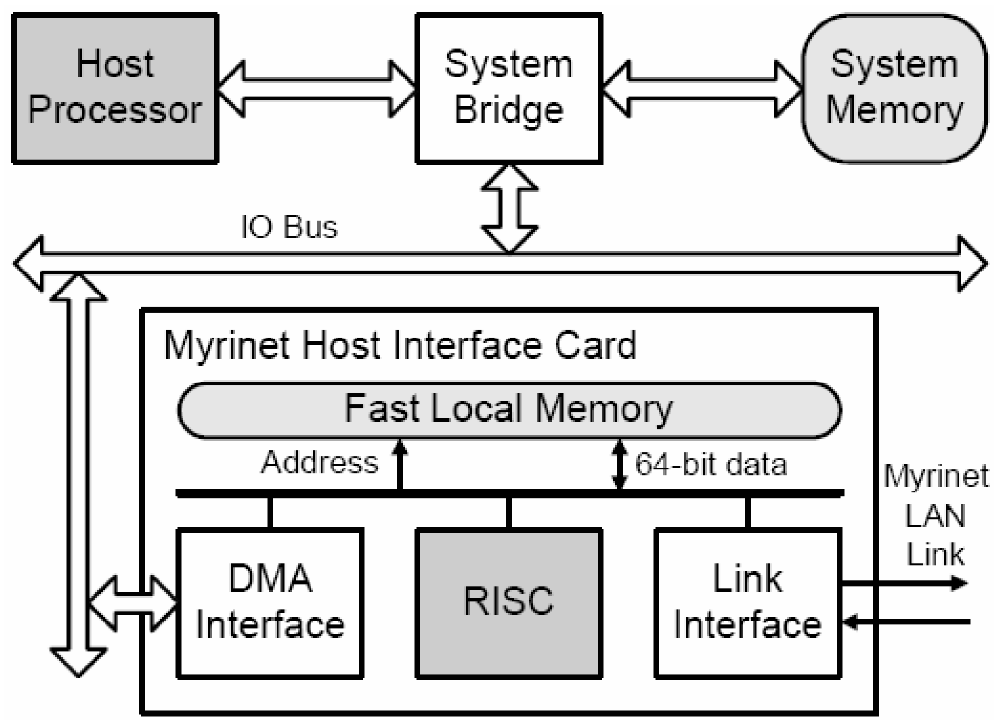
Myrinet: An Example High-speed Network Interface

- A cost-effective local area network technology
  - High bandwidth: ~2Gb/s
  - Low latency: ~6.5 μ s
- Components in an example Myrinet LAN:



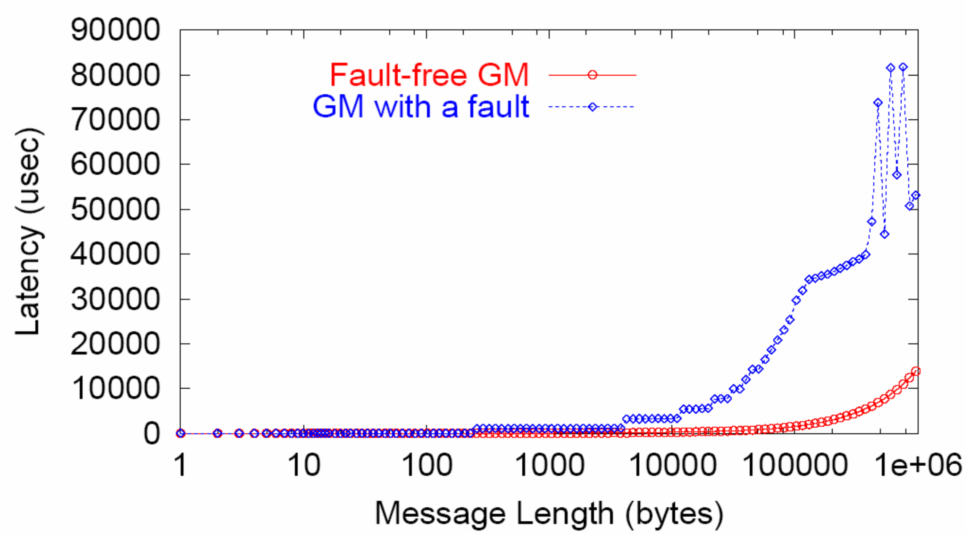
Simplified Block Diagram of The Myrinet Network Interface

- Instruction-interpreting RISC processor
- DMA interface
- Link interface
- Fast local memory (SRAM)

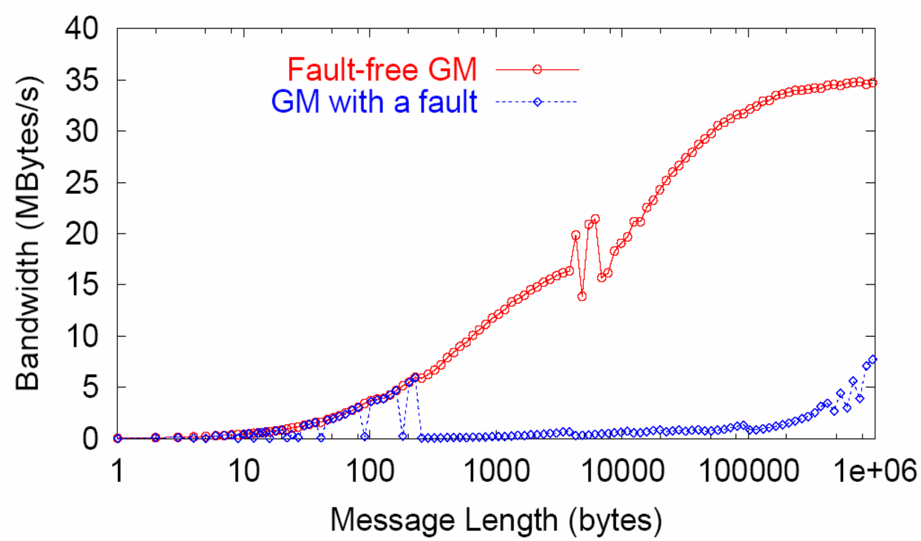


Network Interface Failures

- Transient faults in the form of random bit flips in the network interface
- Failures observed:
  - Network interface hangs
  - Send/Receive failures
  - DMA failures
  - Corrupted control information
  - Corrupted messages
  - Unusually long latency



(a)



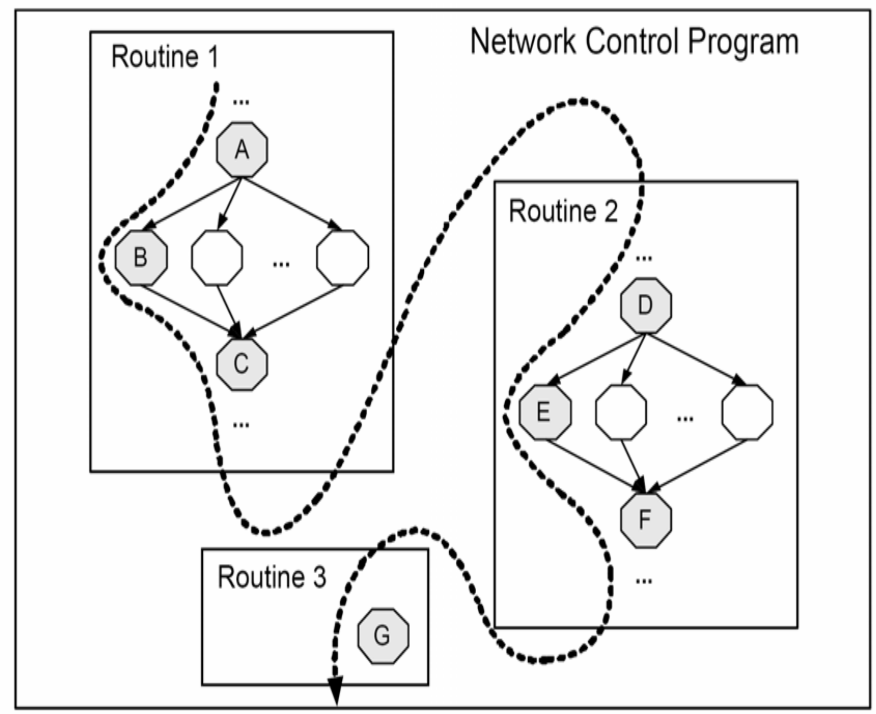
(b)

Failure Detection Strategy

- Interface hangs
  - Software watchdog timer
- Other failures
  - A useful observation: applications generally use only a small portion of the NCP
    - *Directed Delivery:* used for tightly-coupled systems, allows direct remote memory access
    - *Normal Delivery:* used for general systems, allows reliable ordered message delivery
    - *Datagram Delivery:* delivery is not guaranteed
  - Adaptive Concurrent Self-Testing (ACST)
    - Test only part of the NCP
    - Avoids testing & signaling benign faults
    - Can detect hardware & software failures

Logical modules

- Identify the "*active*" parts
- Logical module:
  The collection of all basic blocks that might participate in providing a service
- To test a logical module:
  Trigger several requests/events to direct the control flow to go through all its basic blocks

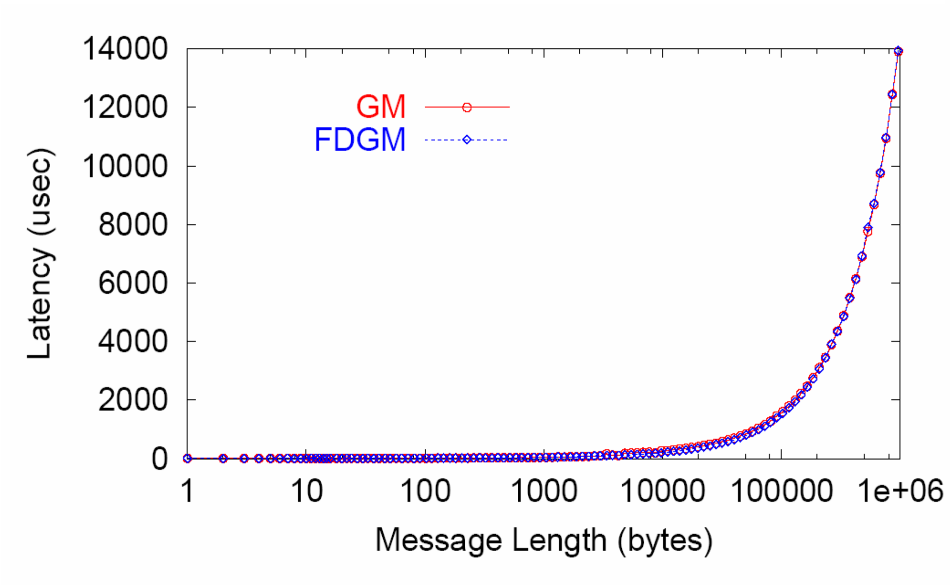


Experimental Results: Failure Coverage

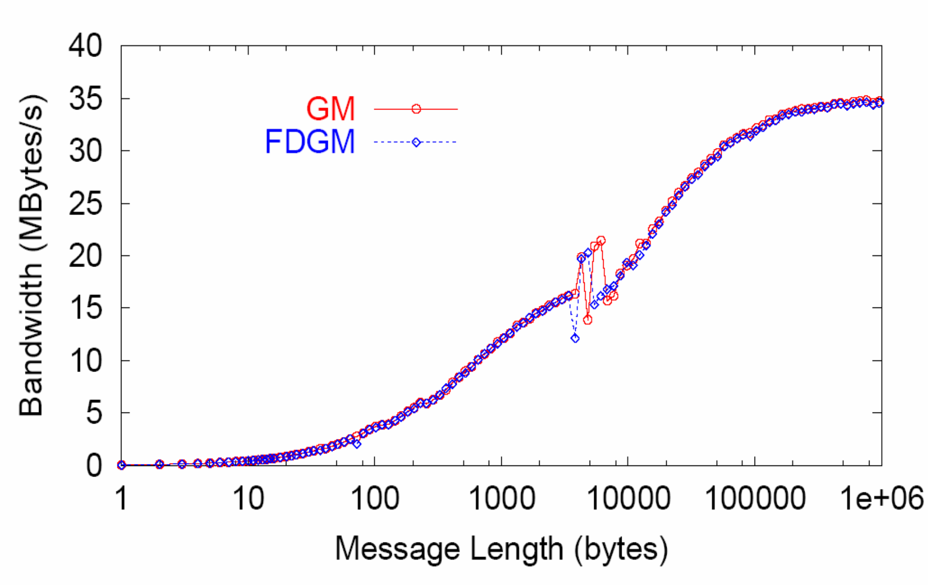
- Exhaustive fault injection into a single routine: send_chunk
- Exhaustive fault injection into special registers
- Random fault injection into the entire code segment

	Coverage	No impact
Routine: send_chunk	99.3%	60.3%
Registers	99.2%	32.3%
Entire code segment	95.6%	93.9%

Performance Impact



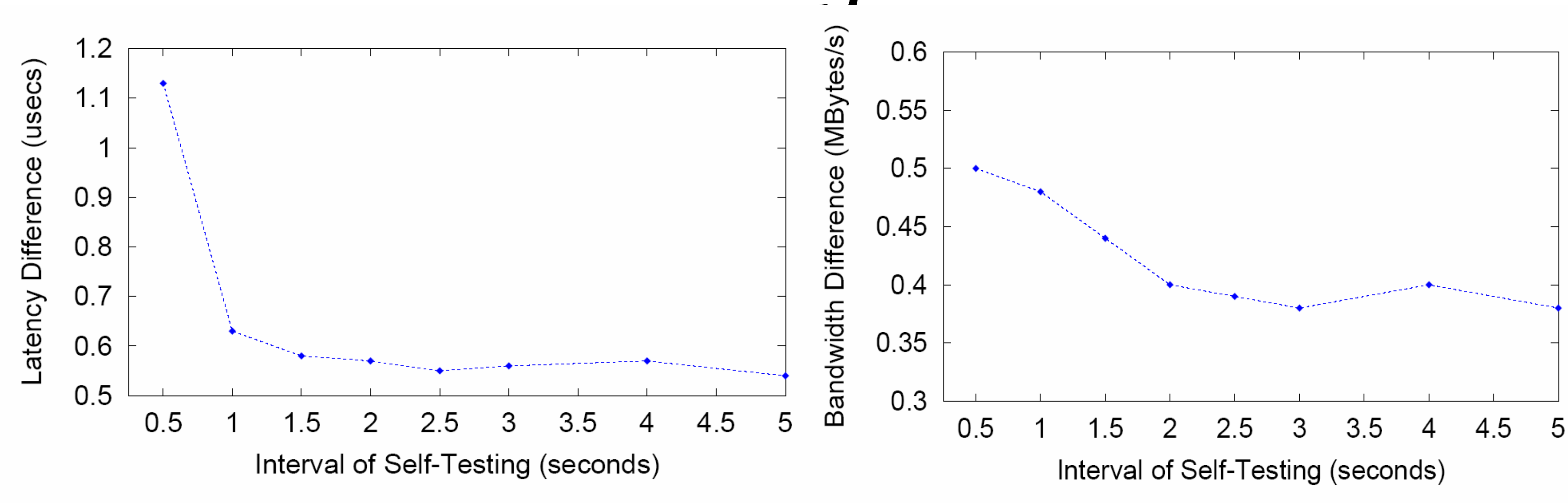
(a)



(b)

- The original Myrinet software: GM
  The modified Failure Detection GM: FDGM
- The MCP-level self-testing interval is set to 5 seconds

Performance Impact For Different Self-Testing Intervals



(a) (b)

- Message length is 2KB
- For the half-second interval
  - bandwidth is reduced by 3.4%
  - latency is increased by 1.6%

Conclusion

- The proposed ACST tests only active logical modules
- Failure coverage: over 95%
- No appreciable performance degradation
- Transparent to applications
- The basic idea is generic – applicable to other fast network interfaces