EECS 571 Principles of Real-Time Embedded Systems
Lecture Note #19:

Real-Time Communications in Point-to-Point Networks

Kang Shin (kgshin@eecs.umich.edu)

Point-to-Point Networks

- Attractive because of fault-tolerance capability
- Allow multiple conversations to go on simultaneously on different links
- Access to the links can be controlled easily
- Drawback: Higher latency
- Desirable: efficient broadcast algorithms

Guaranteed Delivery

- Message generation characteristics
  - Source, Destination
  - Maximum message length: S_{max} (bytes)
  - Minimum inter-arrival time: R_{max} (msgs/sec)
  - Maximum burst size: B_{max} (msgs)
  - Desired bound on message latency: D
- In any time interval of length t, the number of messages generated may not exceed $B_{max} + t \cdot R_{max}$.
- A pair of *uni-directional* real-time channels should be established between source and destination *before* messages can be transmitted on them.

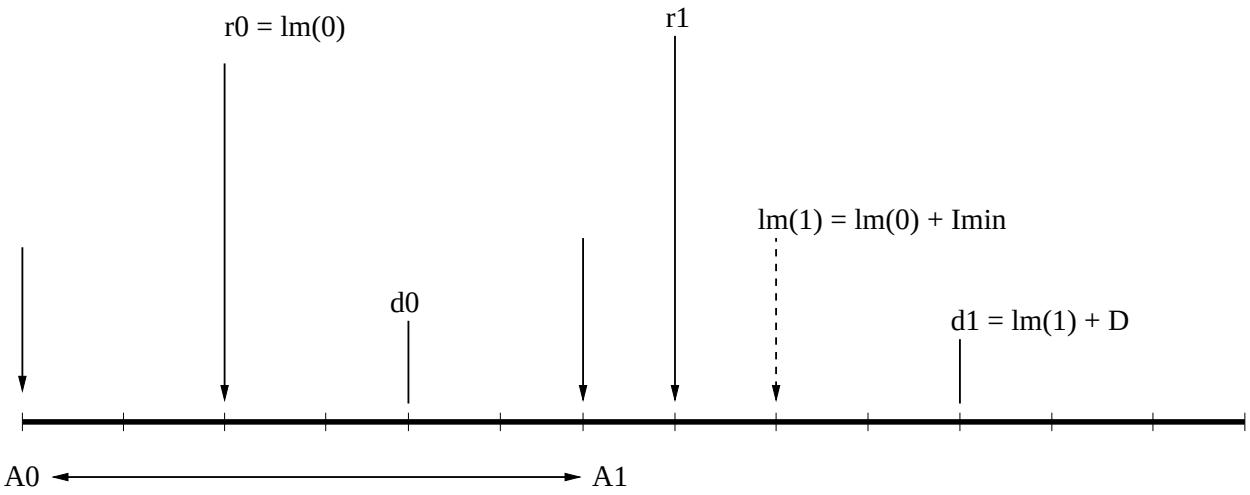
Delivery Time Guarantee

- The *logical generation time*, $\ell(m)$, for a message m is defined as

$$\begin{aligned} \ell(m_0) &= t_0 \\ \ell(m_i) &= max\{\ell(m_{i-1}) + I_{min},\ t_i\} \end{aligned}$$

  where t_i denotes the actual generation time of message m_i, and I_{min} is the reciprocal of R_{max}.
- If D is the end-to-end delay for the channel, the system *guarantees* that any message m_i will be delivered to the destination node by time $\ell(m_i) + D$.

Illustration



Channel Establishment

- Select a source to destination route for the channel
- Compute a feasible worst-case delay at each link (if possible) based on the characteristics of all channels in the system
- Check whether the total delay is acceptable and redistribute the delays
- Compute the buffer requirement at each link

  Note: This computation is dependent upon the link message scheduling algorithm used during transmission.

Delay Computation

- should maintain the feasibility of existing channels
- should obtain the minimum feasible delay
- should be linked to the run-time message scheduling policy
- distinguish between the feasibility testing and the run-time message scheduling policy
- use an optimality result proved by Dertouzos (’74)

Assignment Procedure

- Arrange the channels in ascending order of their associated delay d_i.
- Assign the highest priority to the new channel M_{k+1}. Assign priorities to the other channels based on their delay.
- Compute the new (worst-case) response times r_i' for the existing channels based on this priority assignment.
- Find the smallest position q such that $r_i' \leq d_i$ for all channels with priority less than q.
- Assign priority $q + 1$ to the new channel and compute the response time r_{k+1}'.

Response Time

Consider a set of channels $\{M_i = (C_i, d_i, p_i), i = 1, \ldots, m\}$ which share a common link ℓ.

$$S_i = \{d_i\} \bigcup \{kp_j \mid j = 1, .., i-1; k = 1, .., \lfloor (d_i/p_j) \rfloor\}$$

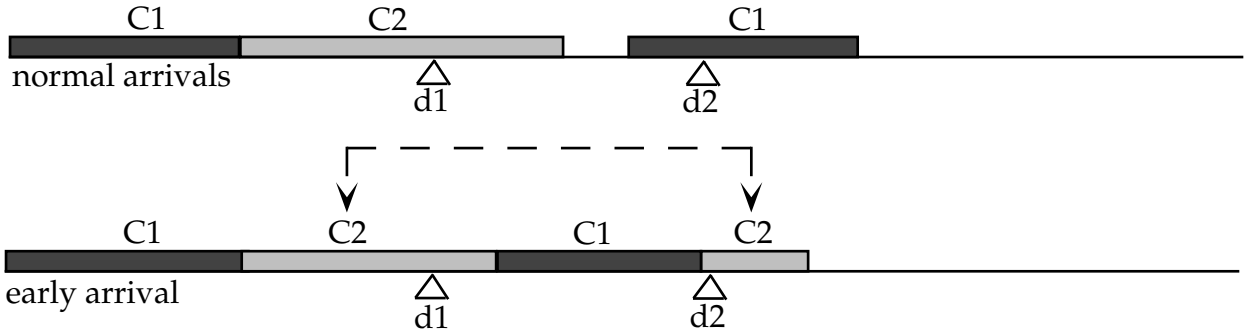
$$W_i(t) = \sum_{j=1}^{i-1} C_j \cdot \lceil t/p_j \rceil + C_i$$

The worst-case response time for messages belonging to M_i is the smallest value of t such that $W_i(t) = t$.

Run-time Scheduling

- Problem with fixed-priority scheduling: arrivals are not strictly periodic
  - arrival time at a node depends on the actual delay at the previous node
  - model allows burst arrivals
- High priority arrivals can disrupt the scheduling of lower priority messages

Illustration



Kang Shin (kgshin@eecs.umich.edu)

Run-time Scheduling

- Deadline Scheduling can be used to overcome this problem
- Based on the logical arrival time of the message at a node

The logical arrival time for m_i at node b, $\ell_{c,b}(m_i)$, is defined as

$$\ell_{c,b}(m_i) = \ell_{c,a}(m_i) + d_{c,a}$$

where $d_{c,a}$ is the worst-case delay for messages on channel M_c at node a.

- Is the feasibility testing still valid?

Run-time Scheduling

Uses a multi-class Earliest Due Date (EDD) algorithm

Queue 1 Packets belonging to real–time channels with $\ell_c(m_i) \leq$ current_time, arranged in the order of increasing deadlines.

Queue 2 Other packets arranged in the order of increasing deadlines.

Queue 3 Packets belonging to real–time channels with $\ell_c(m_i) >$ current_time, arranged in the order of increasing logical arrival time.

Buffer Management

- Buffer space is reserved for channels at the source, destination, and at intermediate nodes.
  - depends on B_{max}, R_{max}, and the link delays
- Flow–control enforced: time-based
  - packets in Queue 3 are considered for transmission only when their logical arrival time $\leq$ current time + *horizon*

Buffer Requirement

- min space $= S_{max} \cdot \lceil (d_{prev} + d_{node})/I_{min} \rceil$
- buf space $= S_{max} \cdot \lceil (d_{prev} + d_{node} + H)/I_{min} \rceil$
- max space
  $= S_{max} \cdot \lceil\ d_{cumul}/I_{min}\ \rceil + S_{max} \cdot B_{max}$

Requirements and Implementation Architecture

Requirements:

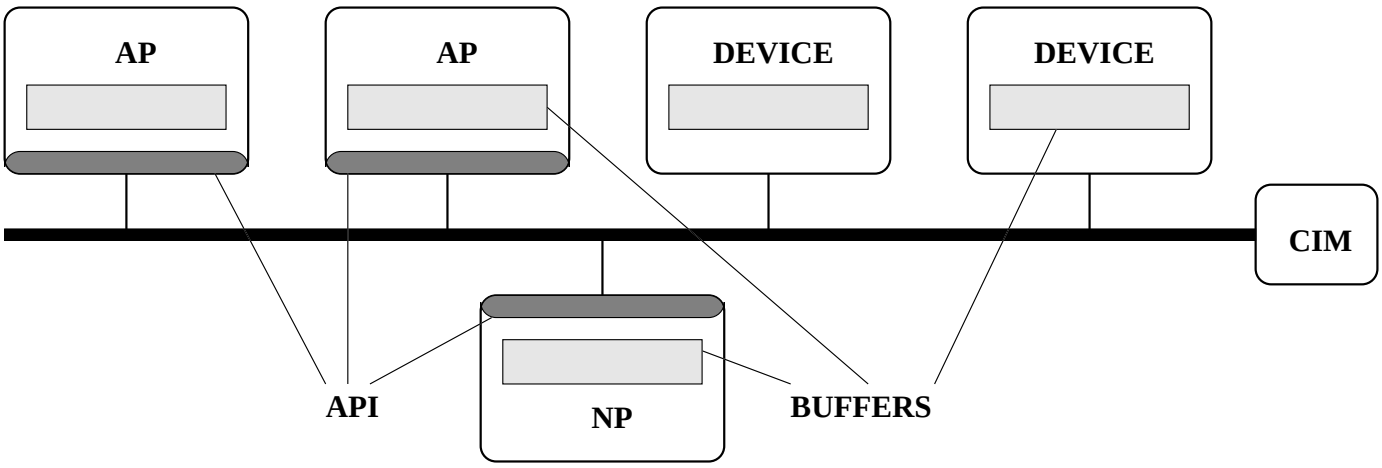
- Setup and teardown of real-time channels
- *QoS-sensitive* data transfer to/from the network.

Shared host resources: bandwidth for bus, link, and protocol processing.
$\Rightarrow$ consumption consistent with relative importance of active real-time channels.

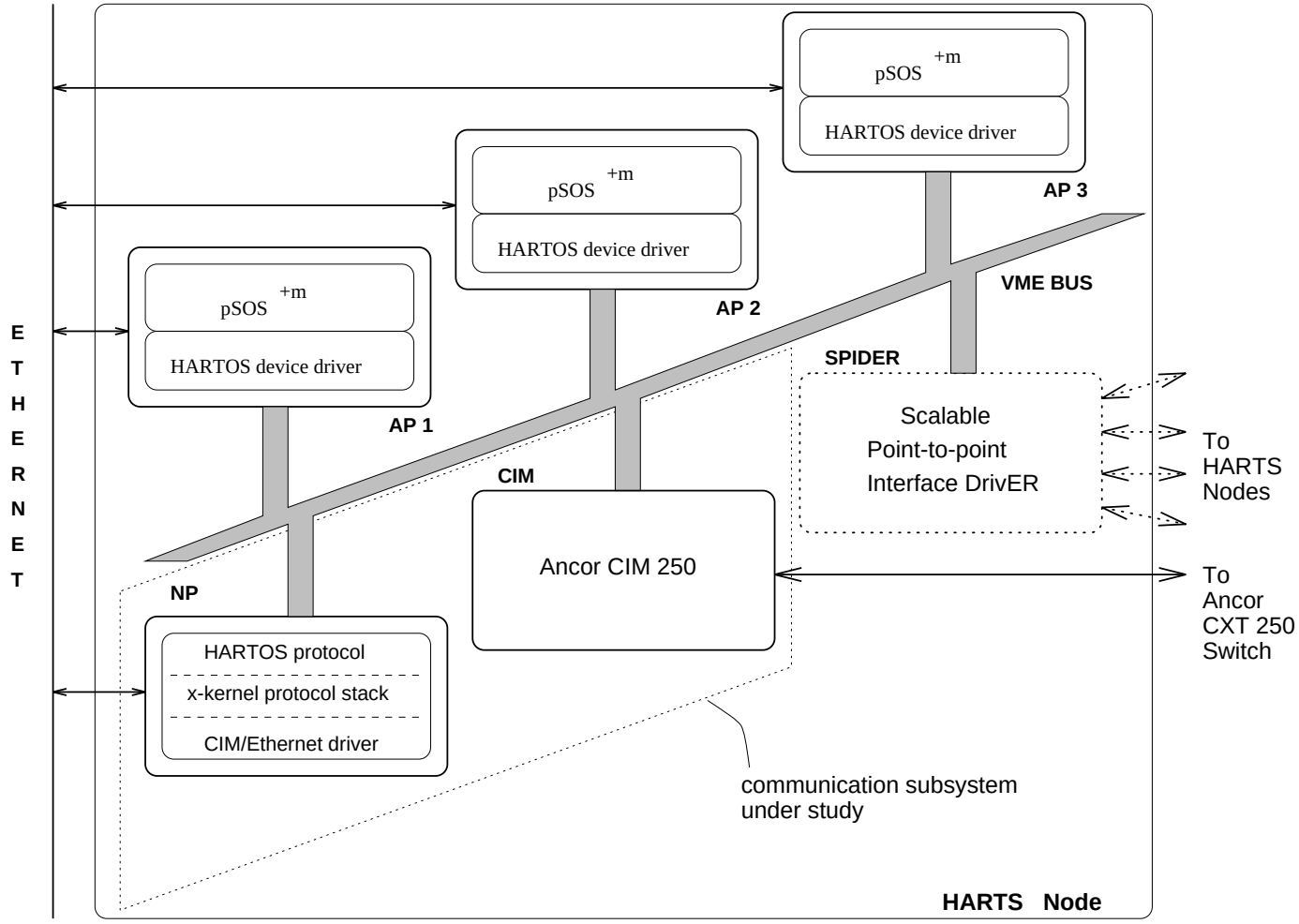
Key architectural features:

- Dedicated protocol processor
- Split-architecture for accessing real-time communication services
- Decoupling of data transfer and control in the communication protocol stack.

Host Architecture

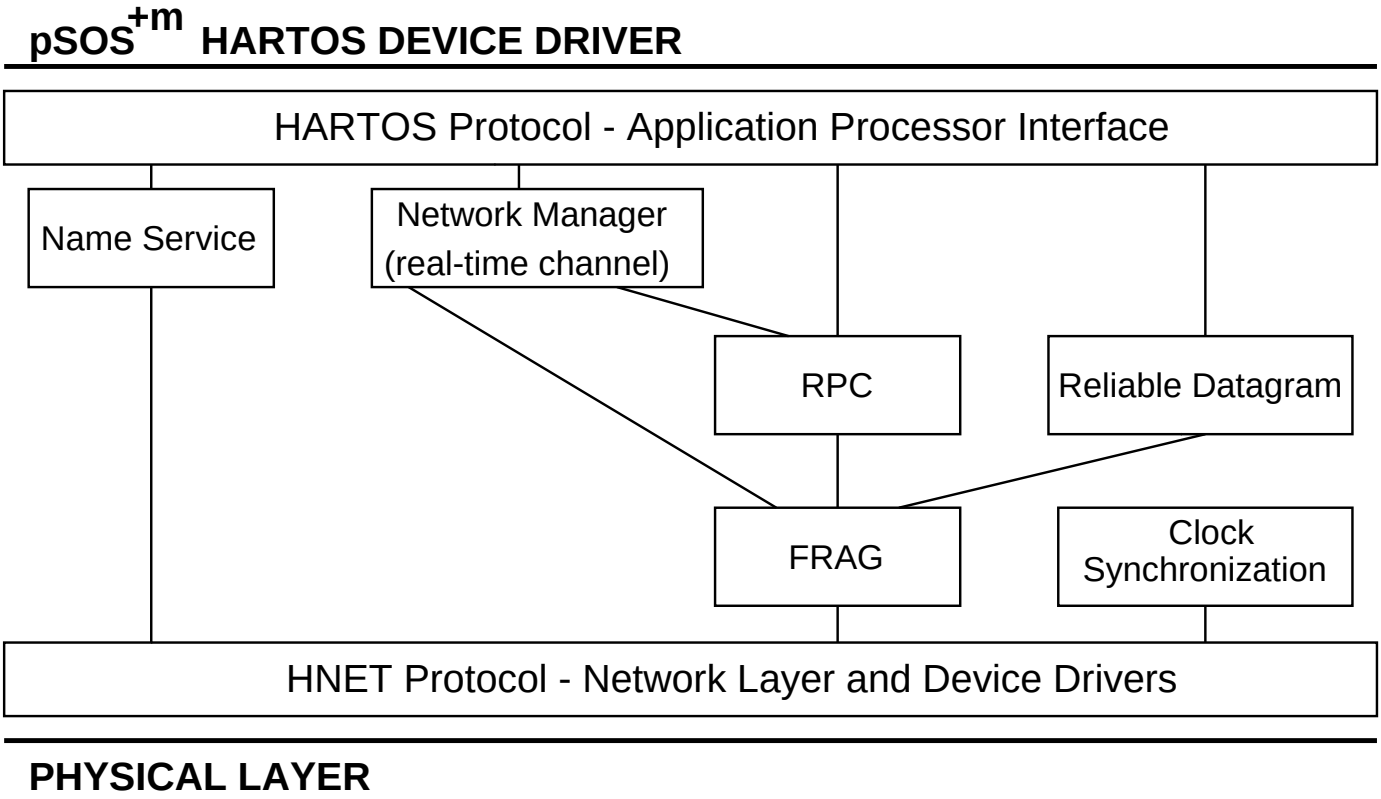


Experimentation Platform



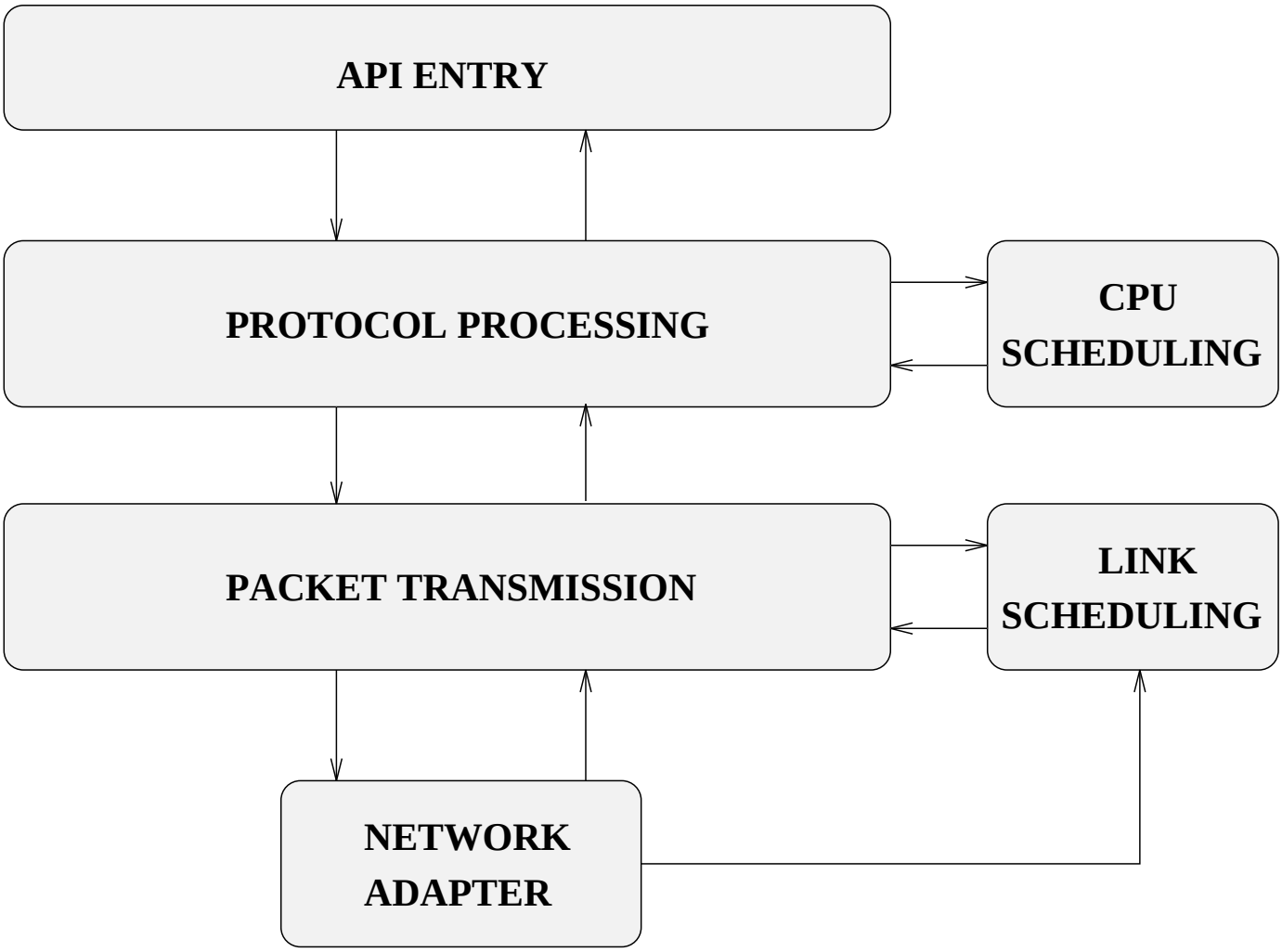
Kang Shin (kgshin@eecs.umich.edu)

Communication Subsystem Protocol Stack

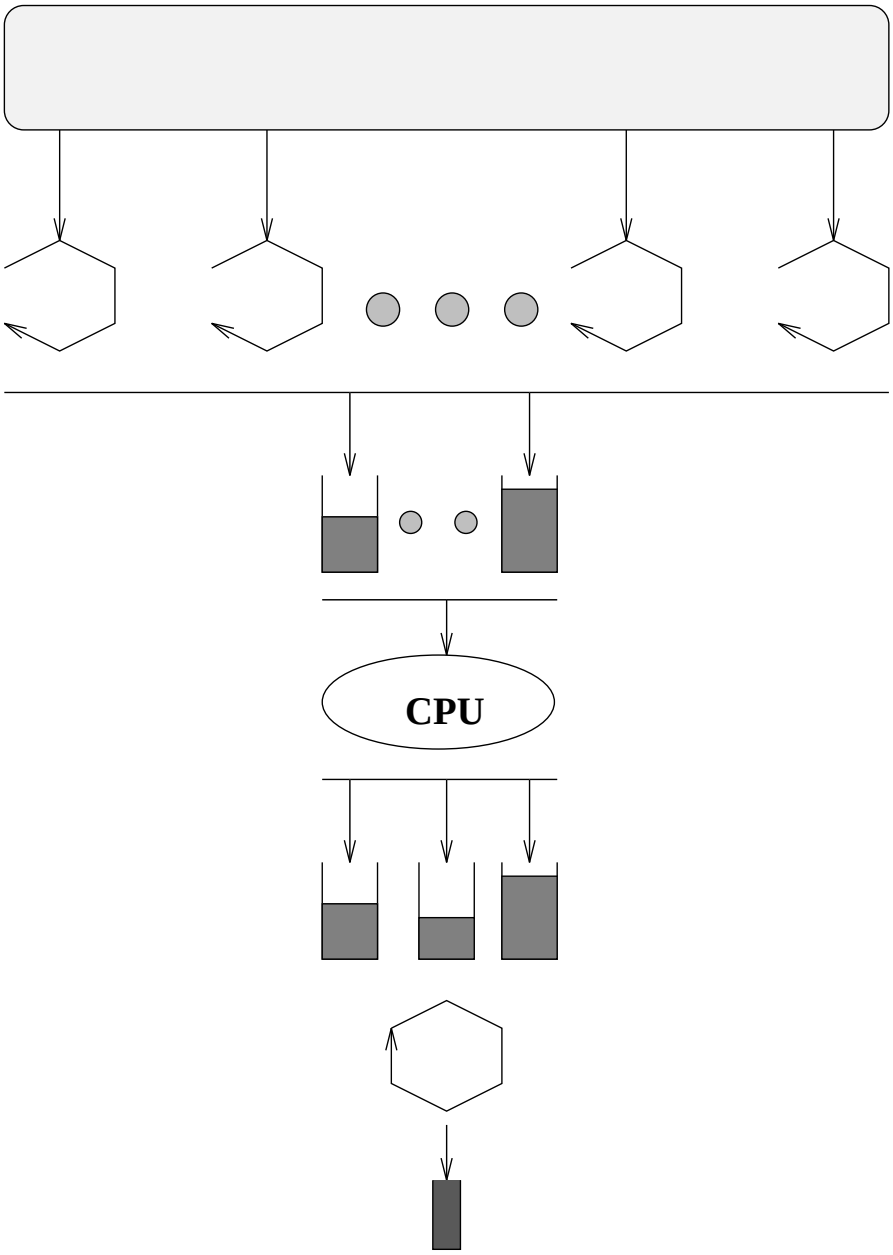


Kang Shin (kgshin@eecs.umich.edu)

Communication Software Structure



Software Architecture on NP



API ENTRY

MESSAGE HANDLERS

RUN QUEUES (FIFOs)

PACKET QUEUES

LINK SCHEDULER

NETWORK ADAPTER

Kang Shin (kgshin@eecs.umich.edu)

Experimental Evaluation of Implementation

How effective is the current implementation in insulating

- real-time traffic from best-effort traffic, and
- well-behaved real-time channels from ill-behaved ones?

Evaluation experiments:

- Effects of best-effort traffic load on real-time traffic
- Effects of burstiness and message size on delay guarantees

Experiments

2-host experiments with measurements at source (transmitting) host.

Traffic sources:

- bursty best-effort “channel”
- bursty real-time channel
- two real-time channels

Processing priority:

- higher priority for processing of real-time traffic
- all real-time channels processed at same priority

Traffic violations: violation of specified message rate

Parameters measured: packet latency, packet queueing delay, and packet loss rate

Summary of Experimental Results

Insulation between best-effort and real-time traffic:

- no real-time packets dropped
- real-time message latencies independent of offered (best-effort) load
- early real-time traffic consumes CPU bandwidth out of turn $\rightarrow$ more jitter
- higher queueing delay for bursty real-time channel
- early real-time traffic does not affect best-effort performance
- best-effort throughput increases with load until system saturates

Summary of Experimental Results (cont'd)

Insulation between ill-behaved and well-behaved real-time channels:

- ill-behaved channels experience significant degradation in performance
- well-behaved channels suffer higher jitter

Inferences:

- more channels or larger messages further exacerbate jitter and delay
- deadline violations may occur

$\Rightarrow$ interference more pronounced with faster medium access latency and faster networks since CPU becomes bottleneck.

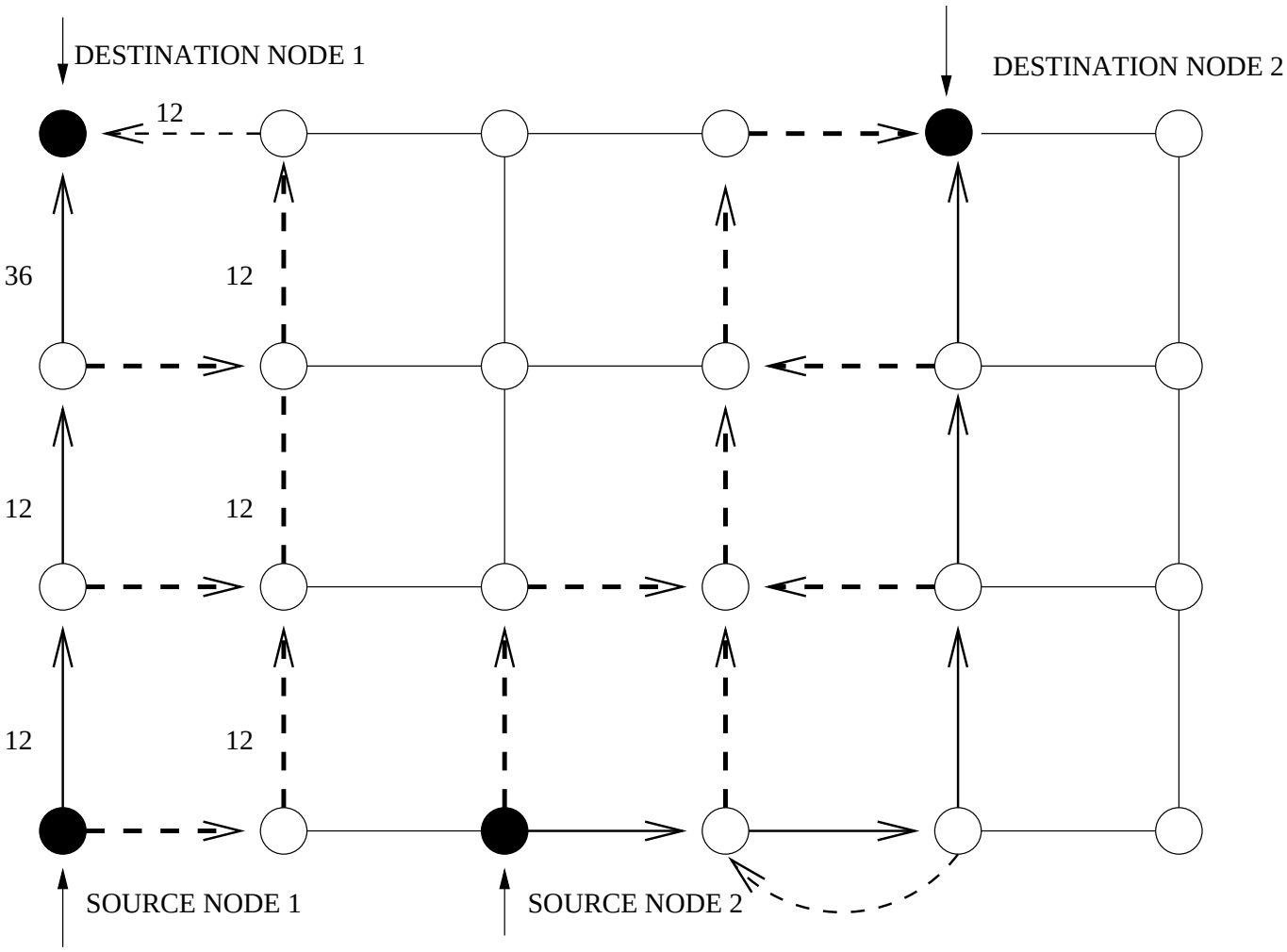
Fault-Tolerant Real-Time Channel

- Static routing makes real-time channels unable to tolerate component failures
- Dynamic routing would make it difficult to guarantee delivery-delay bounds
- Possible solutions:
  - Partially-dynamic routing or local detours
  - Multiplexed backup channels
  - Reactive approach, i.e., do nothing until breaks.

Partially-Dynamic Routing

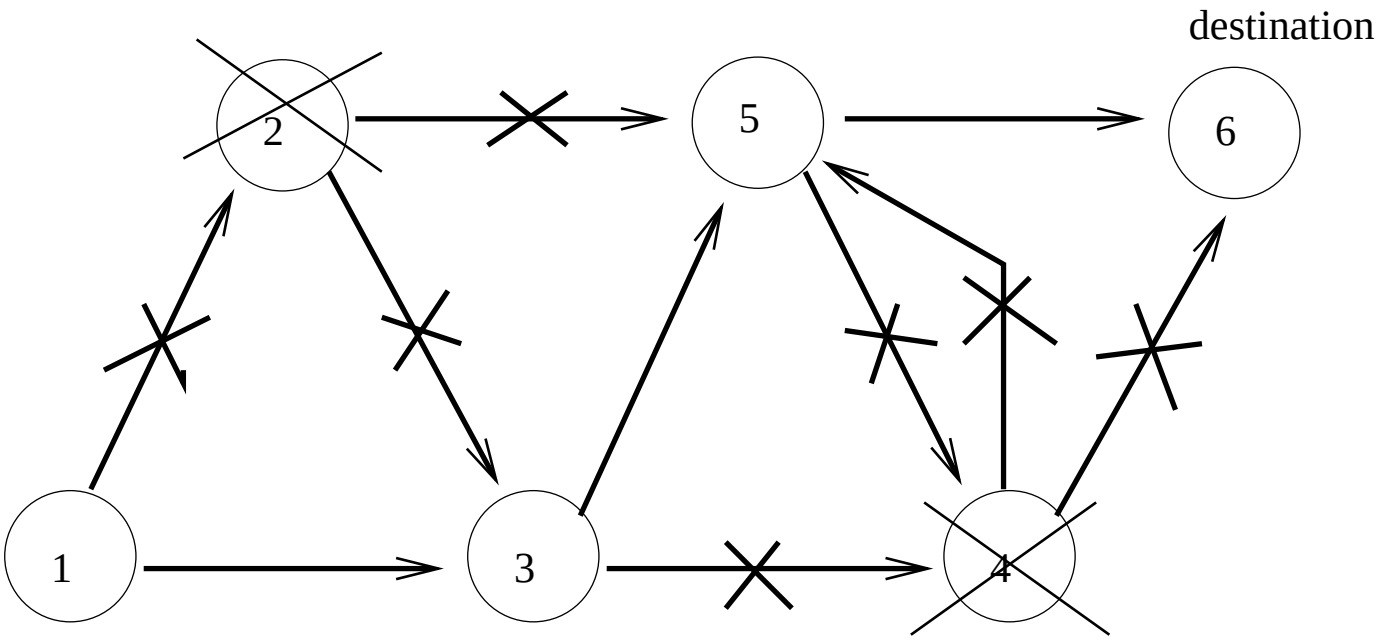
- Set up a primary real-time channel
- Enhance the channel with some extra links and nodes
- Use the primary under normal circumstances, and use the extra links/nodes when the primary breaks down.

Single Failure Immune RTC



Kang Shin (kgshin@eecs.umich.edu)

Isolated Failure Immune RTC



Kang Shin (kgshin@eecs.umich.edu)

Backup Channels

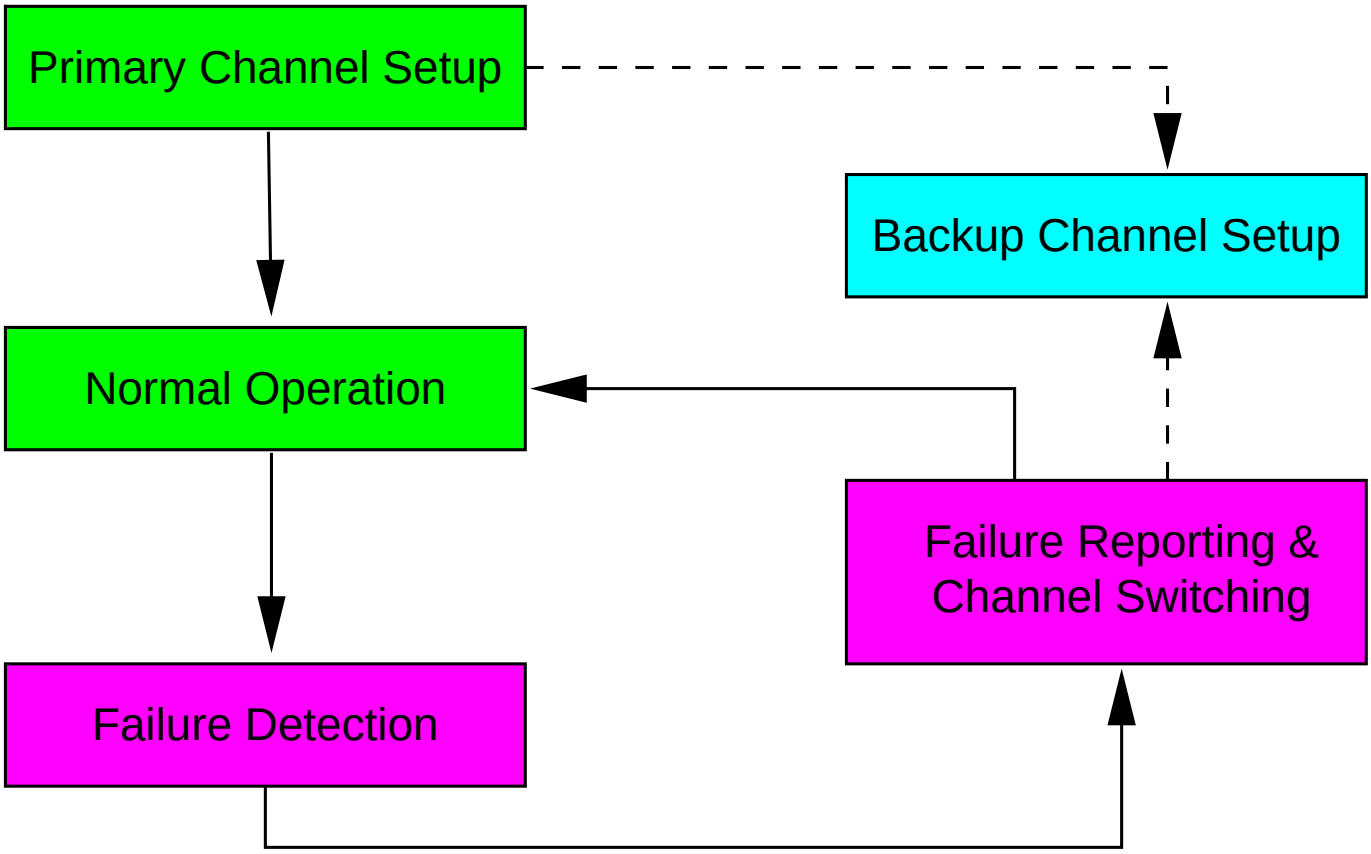
Approach:

- A dependable connection = a *primary* channel + *backup* channels
- Reservation of *spare resources* in advance
- Advance recovery-route selection (off-line end-to-end rerouting)

Issues:

- *Per-connection* dependability-QoS control
- Spare resource alllocation
- Channel failure detection
- Time-bounded failure recovery
- Resource reconfiguration

Overview of Self-Healing Recovery



Kang Shin (kgshin@eecs.umich.edu)

Summary

- E2E real-time communication is achieved via
  - connection establishment
  - run-time scheduling
  - buffer management
- Extensions for fault-tolerance
  - Local detours: SFI and IFI
  - Backup channels and their multiplexing
  - Reactive approach

References can be found from
http://kabru.eecs.umich.edu