3

Fault Tolerant Computing Models and Systems

Classes of failure semantics

- Arbitrary Failures or No Assumptions
- Fail-Silence or Crash
- Weak Fail-Silence or Omissive
- Crash-Recovery

Basic fault tolerance frameworks

- Hardware Fault Tolerance
- Software-Based Hardware Fault Tolerance
- Software Fault Tolerance
- Fault-Tolerant Communication

Classical fault tolerance strategies

- Fault Tolerance versus Fault Avoidance in HW-FT
  - tradeoff between reliable but expensive components and less performant and more complex mechanisms
- Tolerating Design Faults
  - going beyond HW-FT, helpless with common-mode faults (e.g. SW)
- Perfect Non-stop Operation?
  - when no perceived glitch is acceptable

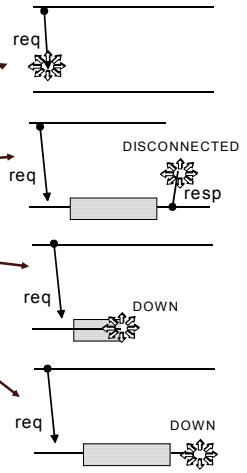
Classical fault tolerance strategies

- Reconfigurable Operation
  - less expensive, when a glitch is allowed
- Recoverable Operation
  - cheap, when a noticeable but acceptable service outage allowed
- Fail-Safe versus Fail-Operational
  - safety track--- when a fault cannot be tolerated, two hypothesis: shutdown, or contingency plan for degraded op. mode

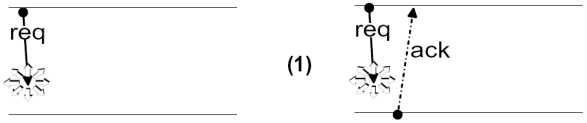
Distributed Fault-Tolerant Computing models

Reliability of remote operations

- more complex than meets the eye…
- remote operation expects a reply to the request
  - what if request is lost?
  - what if reply is lost?
  - what if server dies in the middle?
    - (a) before processing request
    - (b) after, but before sending reply
- these situations are indistinguishable, if comm's level has volatile state and no error detection



Reliability of remote operations
(network failures and remedies)



- communication error detection (*ack*):
  - reduce ambiguity to the server

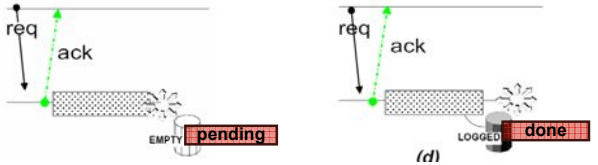
Reliability of remote operations
(network failures and remedies)

- surveillance of communication with server (*aya*):
  - detect communication error in reply

Reliability of remote operations
(server failures and remedies)

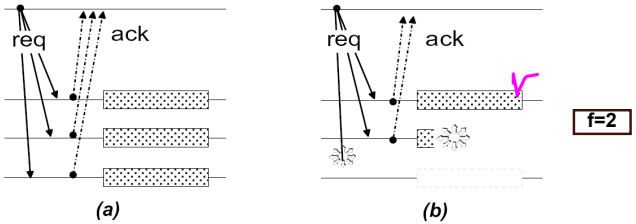
- surveillance of communication with server (*aya*):
  - tell failure from slowness
- server fails before or after executing operation:
  - we cannot know whether or not it was executed
- solution: repeat request until getting a reply (*at-least-once*)?
  - ALO incorrect for non-idempotent operations

Reliability of remote operations
(server failures and remedies)



- solution: in doubt, request just once (*at-most-once*)
- stable memory registers improve AMO :
  - request IDs are stored in disc or NVRAM
  - marked executed or pending
  - when server recovers, it knows what it executed through the end

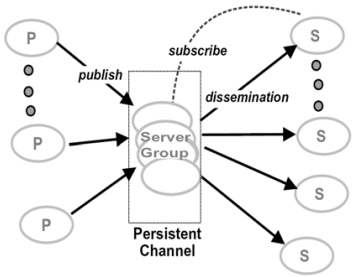
Reliability of event services
(Volatile Channels)



- event diffusion is open loop
  - sender publishes to a group of destinations and goes on
- reliability is given by degree of replication vs. failures
  - request goes to n replicas, first reply is enough
  - request is executed if no more than n-1 failures occur
- request is executed once and only once (*exactly-once*)

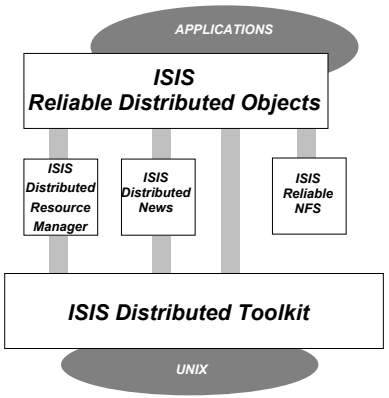
Reliability of event services
(Persistent Channels)

- event diffusion is open loop
  - sender publishes to a group of detinations and goes on
- reliability is given by stable storage
  - channel stores events for later retrieval, so publishers and subscribers do not need to be active at the same time
  - channel is storage media that preserves messages reliably
  - stable storage is F/T namely to crashes, can be implemented as a replicated set of servers

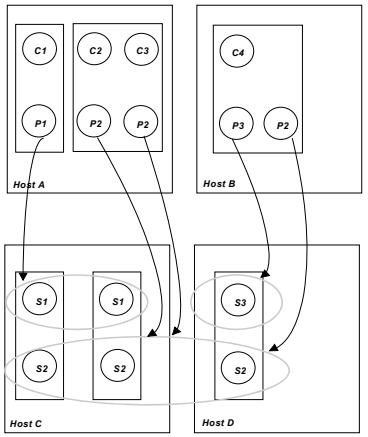


ISIS architecture and functionality

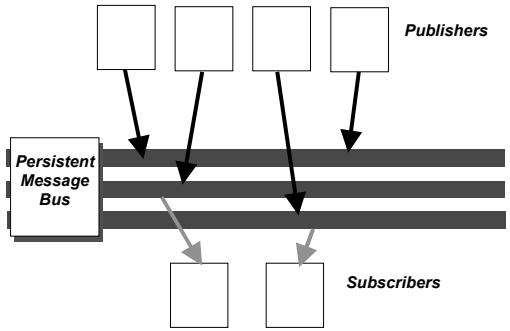
- ISIS Toolkit
  - (basic protocols)
- Reliable Distributed Objects
- Distributed News
  - (publisher-subscriber)
- Reliable NFS
  - (F/T distrib. file system)
- Distributed Resource Manager



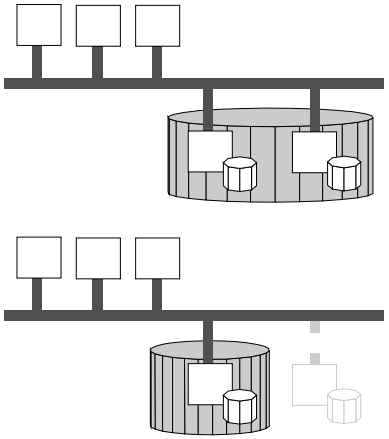
ISIS Reliable Distributed Objects



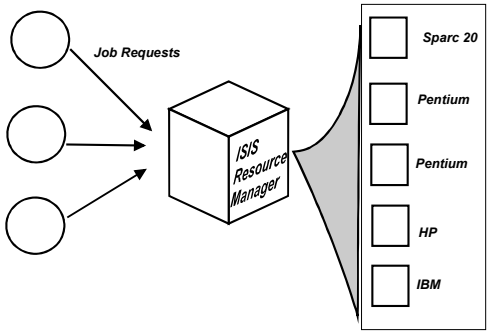
ISIS Distributed News



ISIS Reliable NFS



ISIS Distributed Resource Manager



DELTA-4 System

Definition and Design of a Dependable Distributed Architecture

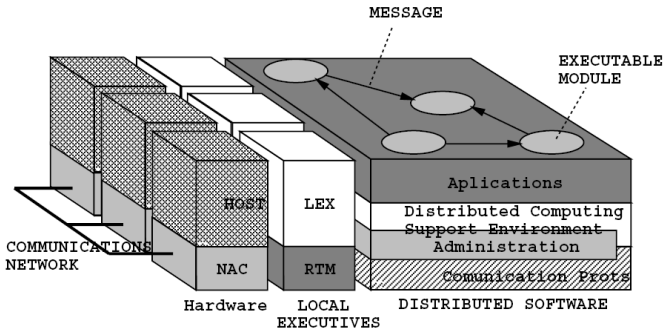
- distributed fault tolerance:
  - profit from geographical separation
  - replicate software, rather than hardware
- performance:
  - high performance intelligent network controllers
  - fast reliable broadcast protocols
- heterogeneity and portability:
  - supports several H/W, LANs and O.S manufacturers
- group-oriented middleware
  - featuring rich communication primitive set
- ODP object model

DELTA-4 System

Definition and Design of a Dependable Distributed Architecture

- open:
  - reliable communication and replication management through OSI-ISO compatible protocols
- versatile application F/T:
  - make *black-* e *white-box* S/W F/T
- communications:
  - reliable and atomic multicast for transparent and distributed replication of S/W components
- dependability:
  - incremental – different types and levels of replication coexist
  - features reliability, availability, security

DELTA-4 Architecture



Distributed and Replicated DBMS

- distributed tac execution
  - TMs address SCh-DM of appropriate node
- advantages:
  - performance and availability of DBMS improves with replication and distribution
- consistency vs. availability
  - pessimistic vs. optimsitic concurrency control
  - strong vs. weak consistency