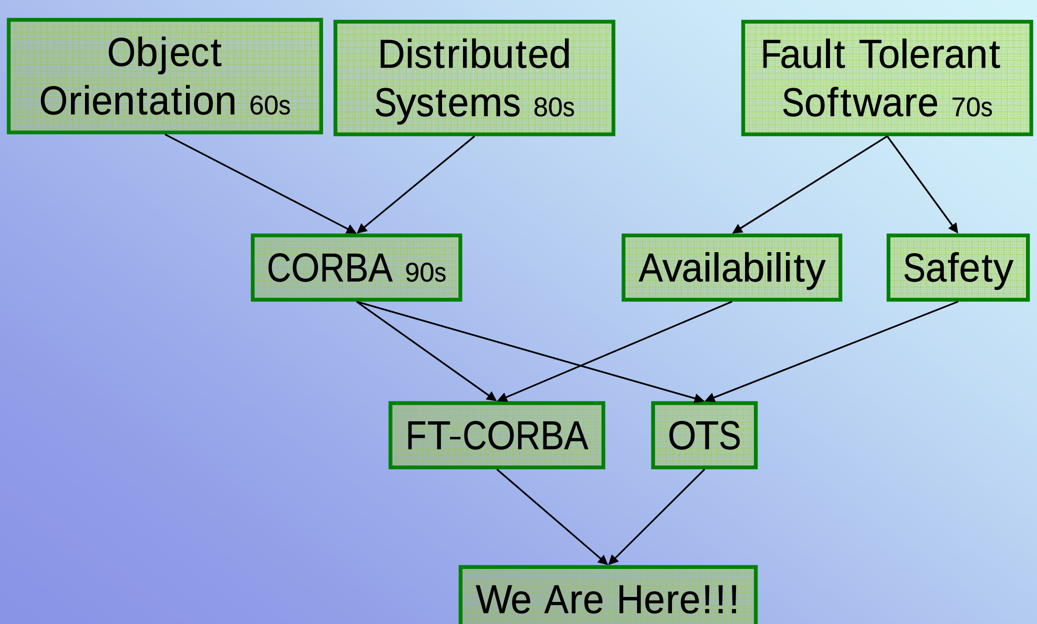
Integrating Transaction and Replication Management Services in CORBA

Hadi Salimi
Computer Engineering Department
Iran University of Science and Technology
Tehran, Iran
h_salimi@mail.iust.ac.ir
January 14, 2006

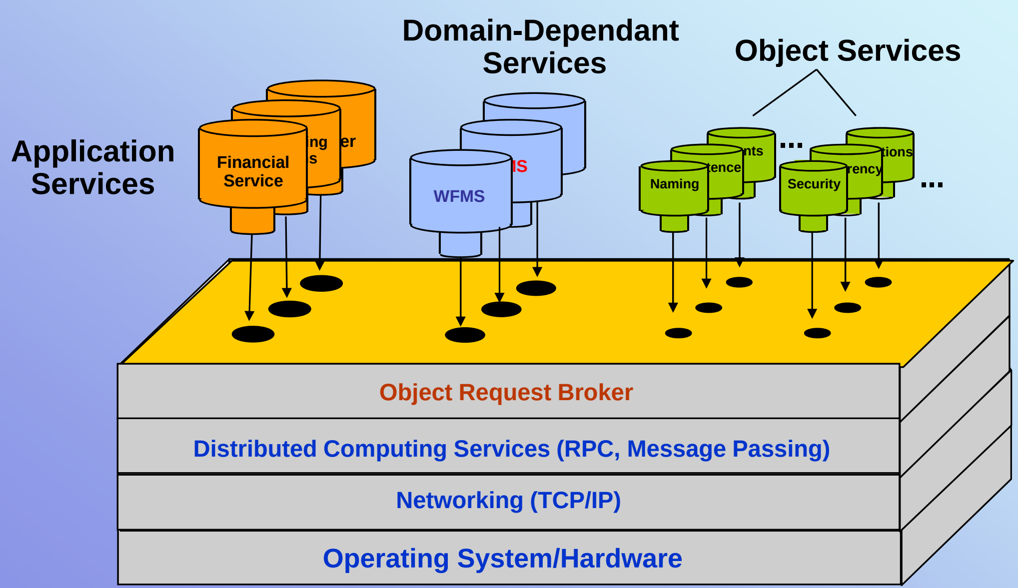
Where Are We?



CORBA

- It stands for Common Object Request Broker Architecture.
- OMG, (a consortium over 800 companies) has produced the CORBA standard.
- CORBA [1] allows objects to invoke services from other objects, hiding differences in location, programming language or platform.

CORBA, A Bird-View

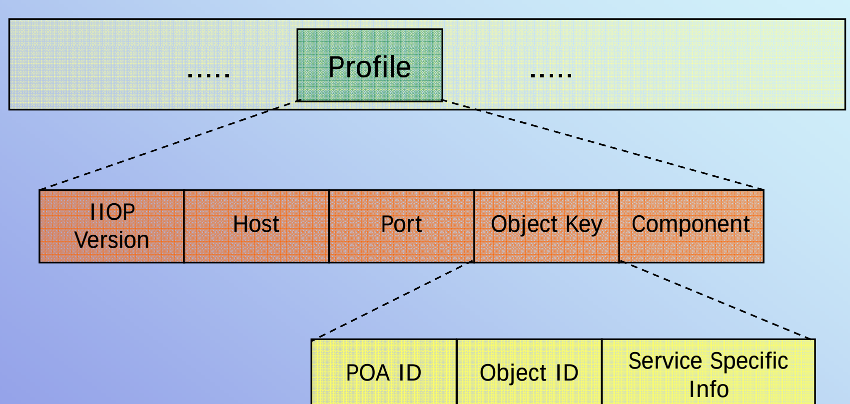


Adopted from [2]

CORBA Features

- Uniform view of the world: everything is an object
- Uniform view of communication: via request invocation
- Communication between objects independent of:
  - Programming languages
  - Physical locations
  - Platform types
  - Networking protocols
- Provides useful Object Services
  - change management, naming, transactions, security, query, etc.

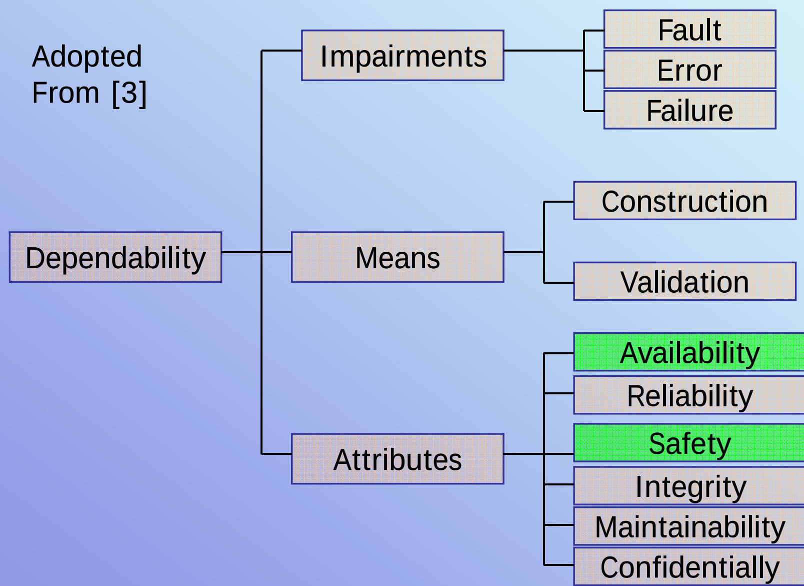
Interoperable Object Reference (IOR)



- IORs are used by the ORB to transparently locate objects

Software Fault Tolerance

Adopted
From [3]

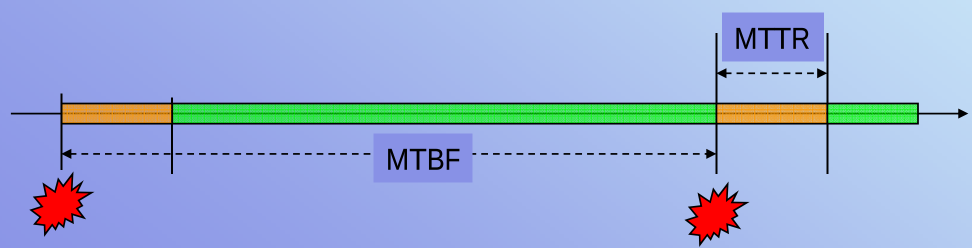


Safety

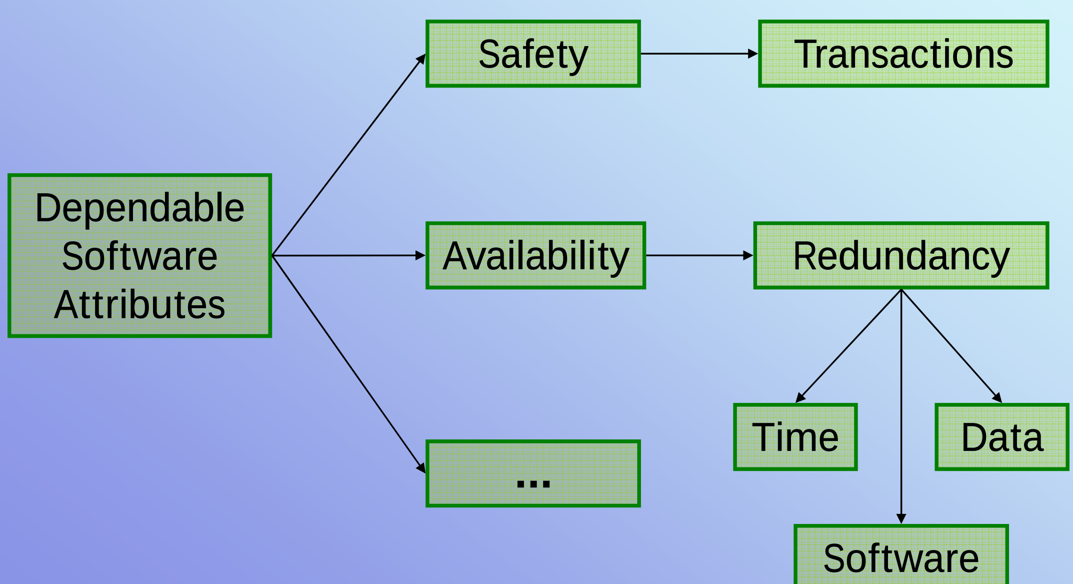
- Software safety can be defined [7] as features and procedures which ensure that the likelihood of an unplanned event is minimized and its consequences are controlled
- Thereby preventing accidental injury or death, whether intentional or unintentional

Availability

- Measure of the time that the system is available for use
- A = MTBF/(MTBF + MTTR) where
  - MTBF is Mean Time Between Failures
  - MTTR is Mean Time to Repair



Software Techniques



Transaction

- It was originally developed in the context of database management systems.
- From a database point of view it is a very elegant way to keep the *data* consistent even in the presence of highly *concurrent* data accesses [4].

There are two mistakes one can make along the road to truth. Not going all the way and not starting.

---Buddha

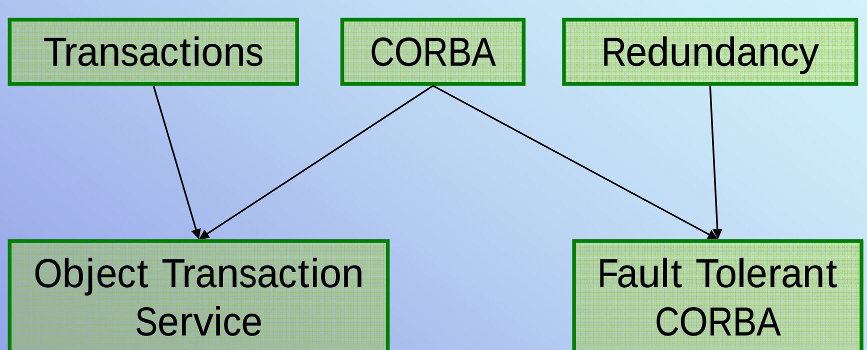
Transaction (cont.)

- Distributed Systems' point of view
  - They allow a process to access and modify multiple distributed resources as a single atomic operation [8].
  - If the process backs out halfway during the transaction, everything is restored to the point just before the transaction started.

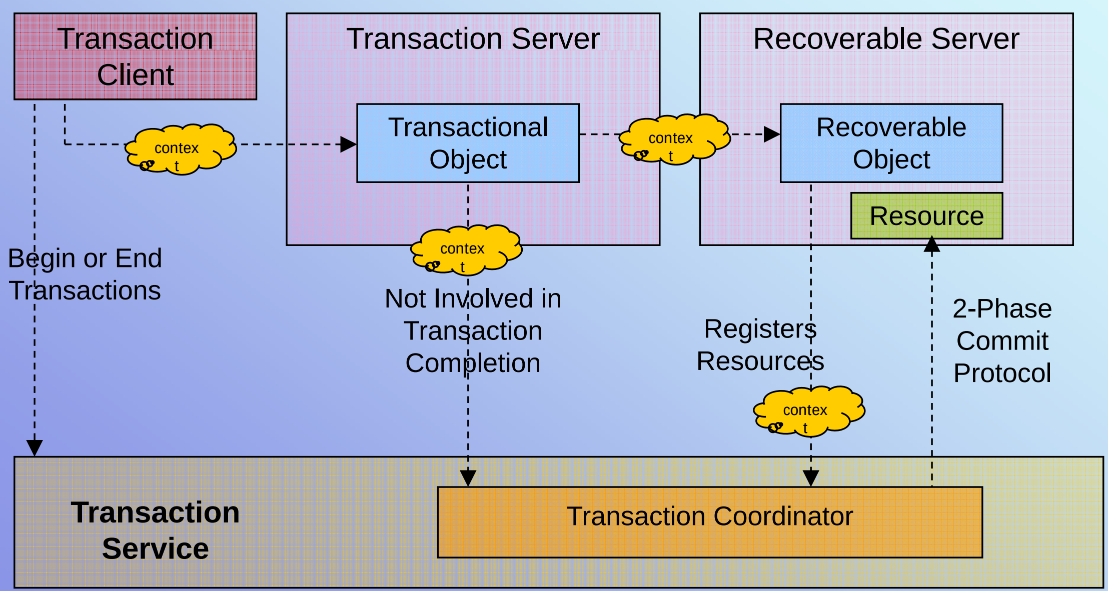
Redundancy

- Software redundancy includes additional programs, modules, functions, or objects used to support fault tolerance [9]
  - Data
  - Time
  - Software

CORBA & FT



OTS Architecture [10]



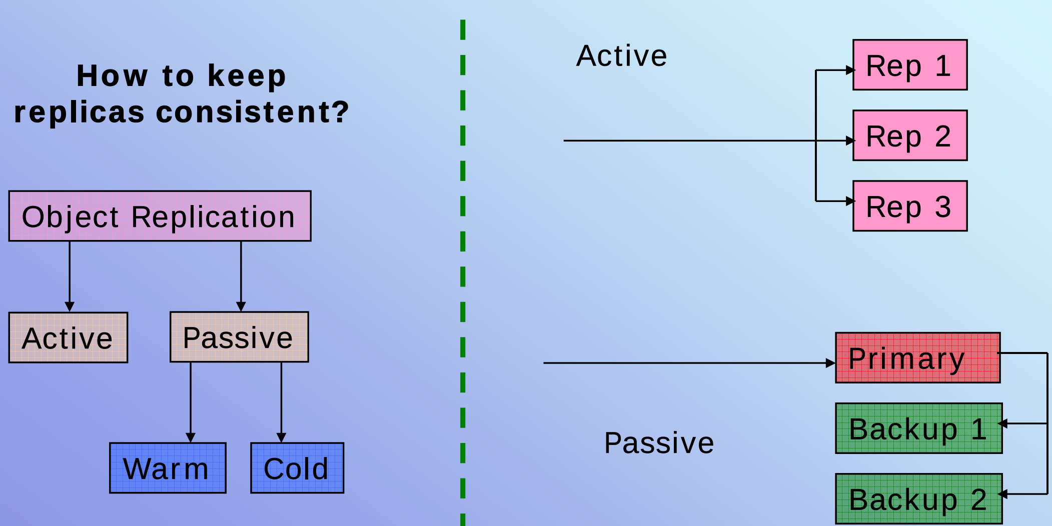
FT-CORBA [11]

- At 1998 OMG issued the RFP
- In early 2000 the first version released
- The last version, December 2001
- Using
  - Replication
  - Object Groups

Replication

- The various approaches to fault-tolerant CORBA are alike on their use of replication [12].
- The behind idea is to mask the failure of an object by making extra objects.
- In the case of a failure, the fault tolerant ORB transparently redirects a failed request to a live replica.

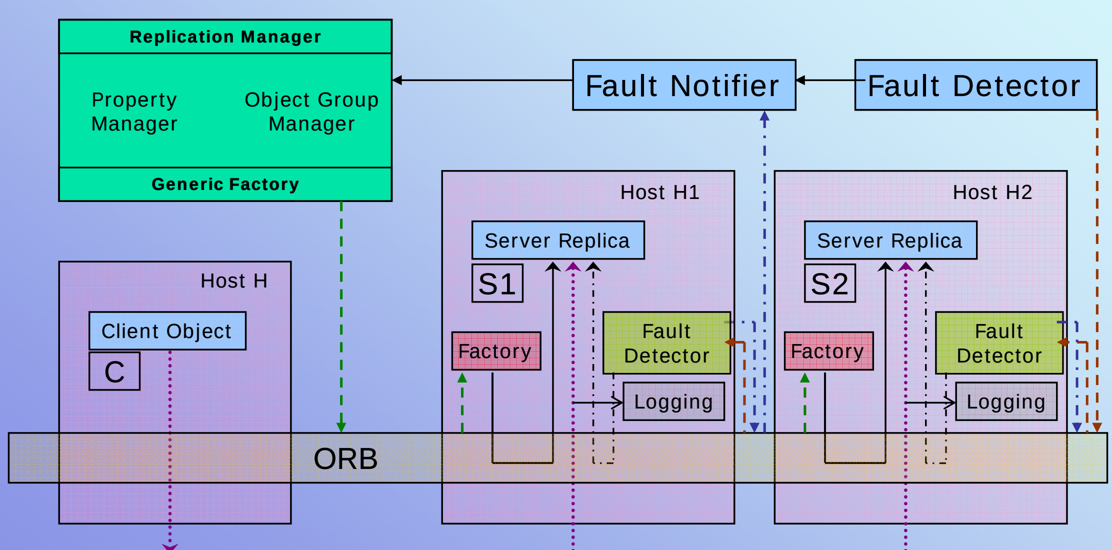
Replication Models



Object Groups

- Object group represents a replicated object and the group members represent the individual replicas of the object.
- Each object group has an Interoperable Object Group Reference (IOGR).
- Using an IOGR, a fault tolerant ORB is capable of accessing each object replica.

FT-CORBA Architecture



FT-CORBA Implementations

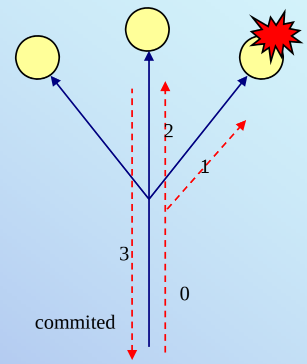
- Delta-4 [13]
  - 1990 – supported by CEC through ESPRIT Project
- Arjuna [14]
  - 1994 – Newcastle University
- Orbix-Isis [15]
  - 1994 – IONA Technologies Co.
- Electra [16]
  - 1995 – Zurich University
- DOORS [17]
  - 1997 - Bell Labs Research
- OGS [31]
  - 1998 – Swiss Federal Institute of Technology
- IRL [18]
  - 1999 – Rome University
- AQuA [30]
  - 1999 – Illinois University
- Eternal [19]
  - 2001 – US Air force research Lab.

Why Integration?

	Replication	**Transaction**
Standard	FT-CORBA	OTS
Tier	Control Tier	Data Tier
Attribute	Availability	Safety
Technique	Roll-forward	Roll-back

Roll-Forwarding

- Replication has brought a new termination style into transaction literature.
- Roll-forwarding means although there are errors during the execution of a transaction, the transaction can be committed safely, just by redirecting the failed request to a fresh replica.



Related Work

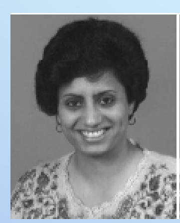
- The new models of replication and transaction are a bit different form their traditional concepts.
- As an example:
  - Concurrency is considered separately.
  - The support of transaction server variety is essential.
  - The granularity of transactional elements is coarser.

Related Work (cont.)

- So, the related works can be categorized into:
  - Integration of traditional concepts
    - Is not interested here, you can refer to [21], [22], [23], [24], [25] and [26].
  - Integration of new concepts
    - Have mainly focused on reconciling replication and transaction in distributed systems, specially the ones that are built my means of CORBA.

Related Work 1

- *Felber* and *Narasimhan* who are pioneers in FT-CORBA, have issued a new publication [27] recently.
- They have indicated that reconciling replication and transaction concepts in CORBA is still an open issue.



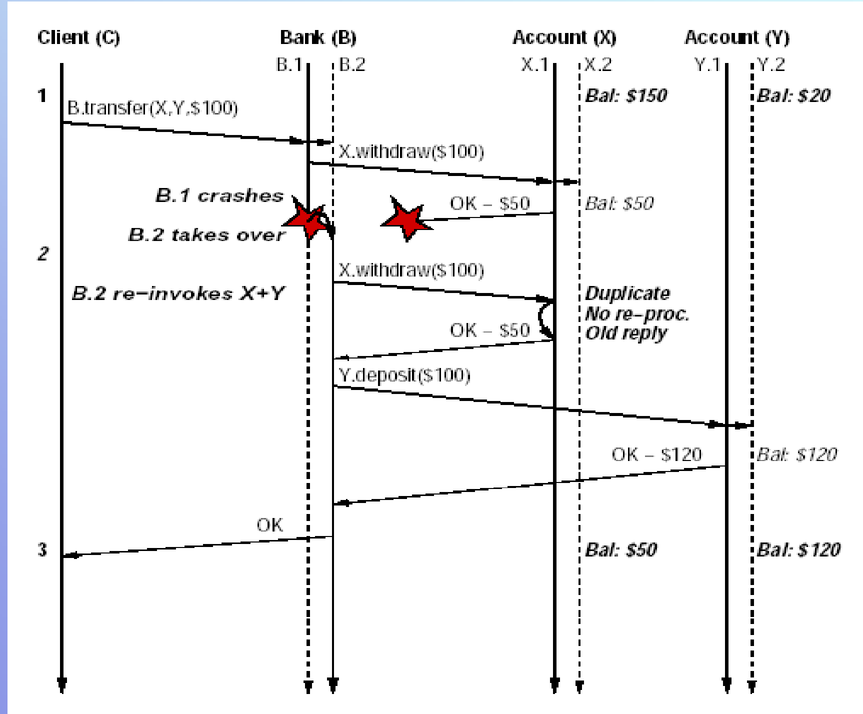
Related Work 2

- Froland and Guerraoui have claimed that [28] the current standards can not be integrated due to some limitations:
  - OTS does not support fine-grained replication.
  - *Duplicated elimination* cannot be guaranteed.
  - *Output determinism* problem.

Related Work 3

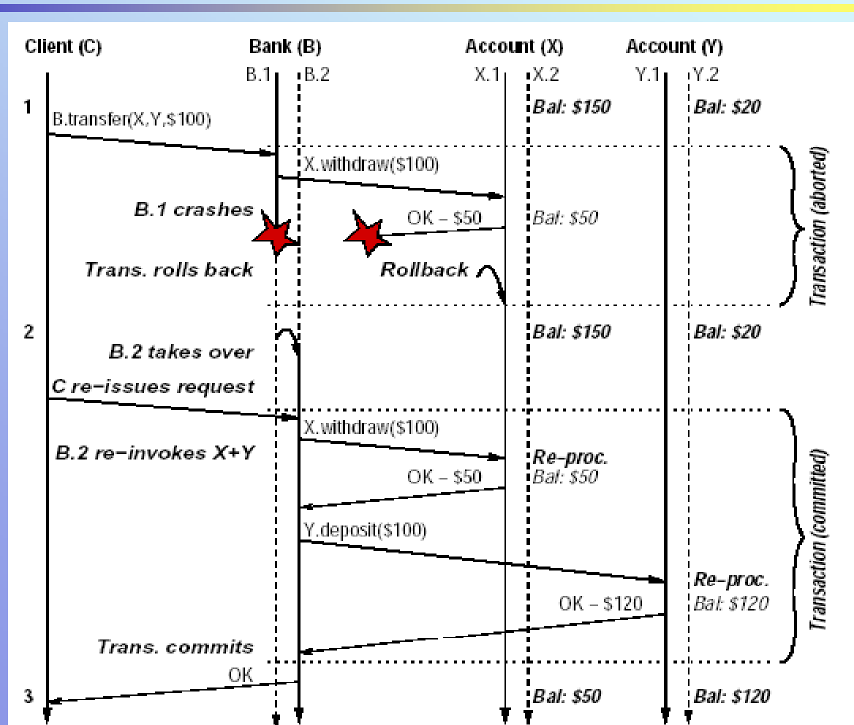
- Felber and Narasimhan have introduced a protocol [20] in order to use transactional operations on replicated objects.
- They have clarified their protocol with a simple bank balance transfer operation example.

Related Work 3 (cont.)



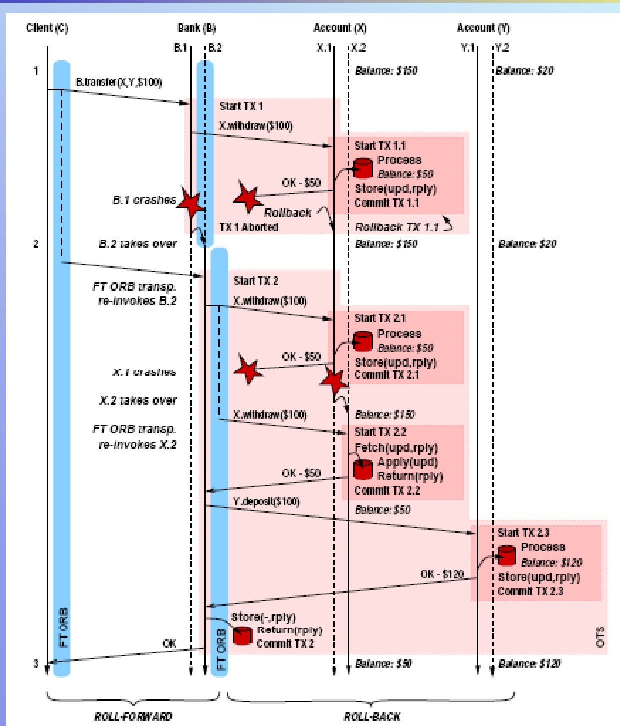
Roll-Forward Approach

Related Work 3 (cont.)



Roll-Back Approach

Related Work 3 (cont.)

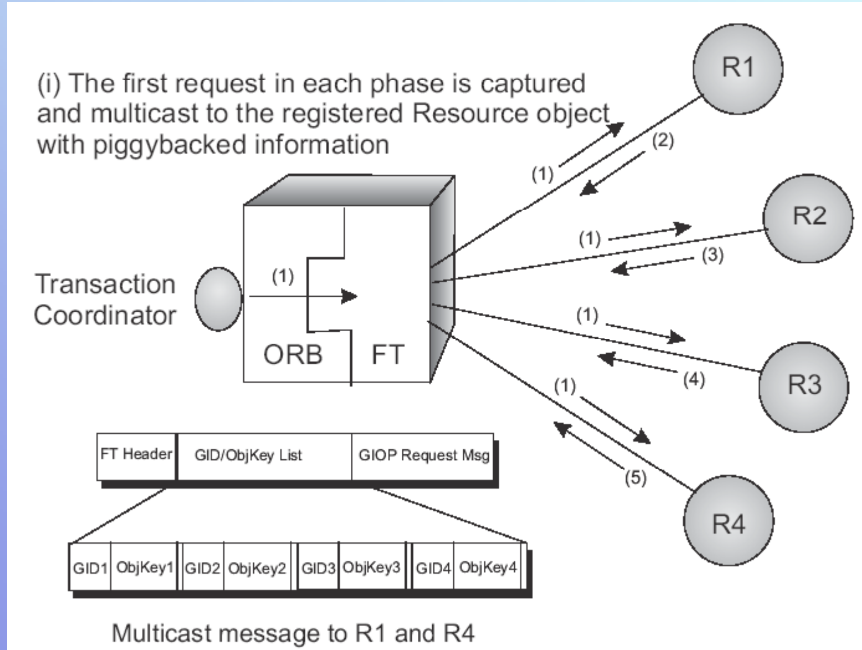


Their proposed approach

Related Work 4

- Zhao and Moser [29] have shown that sending a multicast message to all transaction resources can decrease the overhead of 2-phase commit protocol.

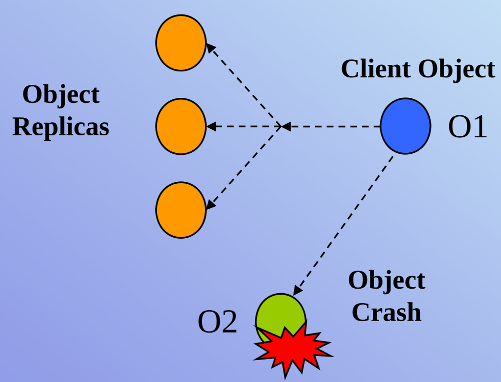
Related Work 4 (cont.)



Why they do not add up?

- Almost all of them suppose that middle tier objects are stateless.
- All of them use total order multicast protocols. The drawback of these protocols will be highlighted.
- All of them suppose the existence of nested transaction.

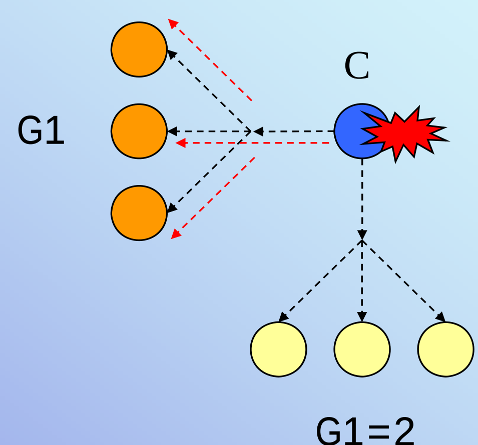
Total Order Multicast Protocols



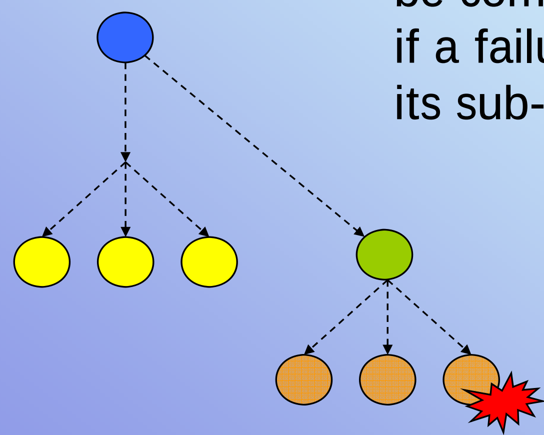
- Suppose that O1 intends to make an atomic call to some objects.
- What will happen if O2 fail during this operation?

Total-Order Multicast Protocols

- What will happen if C fails after updating G1, but just right before updating G2?



Nested Transaction



- A nested transaction can be committed successfully if a failure occur in one of its sub-transactions

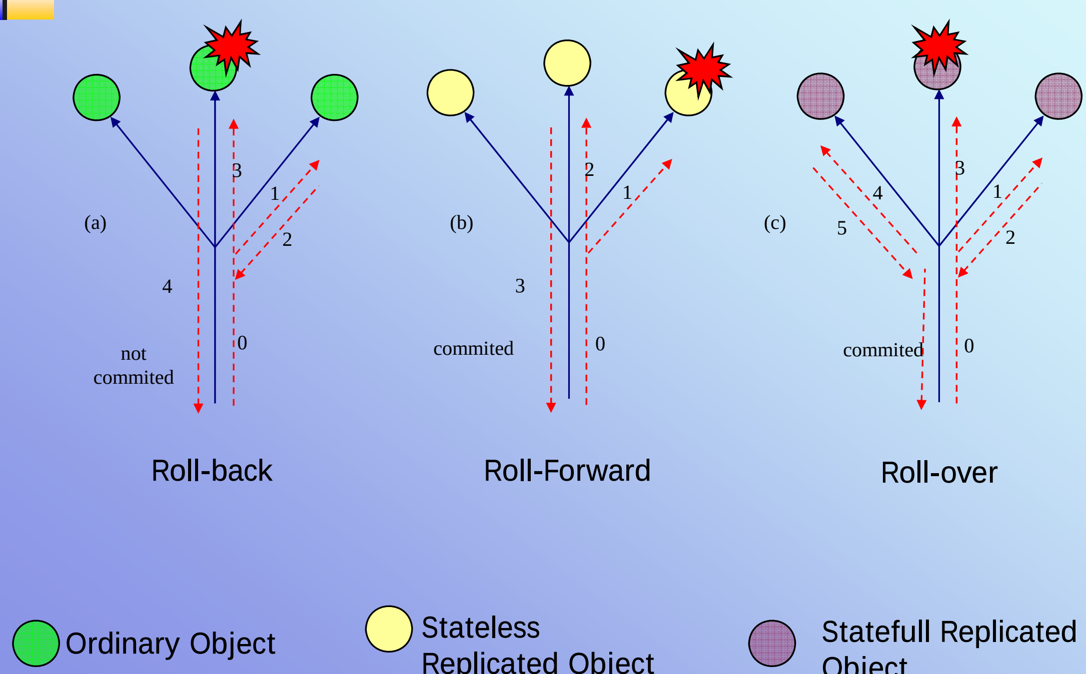
Replication-Aware Transactions (RATs)

- From another point of view, any failure in the scope of a transaction that is executing on a group of replicated objects can be easily ignored by using *Replication-Aware Transactions* (*RATs*):
  - In the case of stateless objects, redirecting a failed request to a live replica is the remedy.
  - But in the case of statefull objects, this technique is not enough.

Replication-Aware Transactions (cont.)

- In the case of statefull objects, omitting the object from its group and recreating it will solve the problem.
- We call this termination style *roll-over.*
- For each object recreation, the state of the object should be transmitted.

Roll back-forward-over

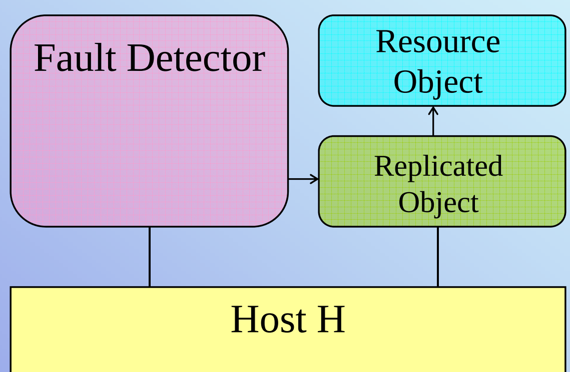


OTS Extension

```
void
ResourceManager::registerResource(
    CosTransactions::Resource_ptr r ,
    const char* name) throw()
{
  ResourceRecord          record;
  Record.name = name;
  //... other initializations for record

  if (strstr(name , "~$Replicated$~"))
          record.setReplicated(true)
  else
          record.setReplicated(false);

  resources_.push_back(record);
}
```

```
CosTransactions::Vote
ResourceManager::prepare()
{
.....

    switch(v)
    {
    case CosTransactions::VoteRollback:
    if (res -> isReplicated())
          resources_.Remove(res);
    else return v;
              //other cases come here
    }
}
```


Fault Detection



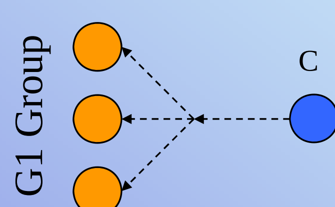
- How to detect the:
  - Crash of H
  - Crash of Replicated Object
  - Crash of Resource Object

Group Failure

- What will happen if all replica objects fail due to a:
  - Network Failure
  - Rare but possible group failure
- In this case, the resource objects that belong to an object group should be distinguishable.

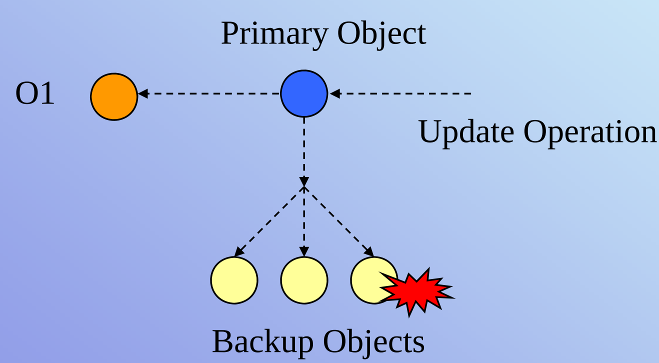
Replication Style Support

- Active Replication:
  - To update a group of active replicated objects, RATs can be helpful.



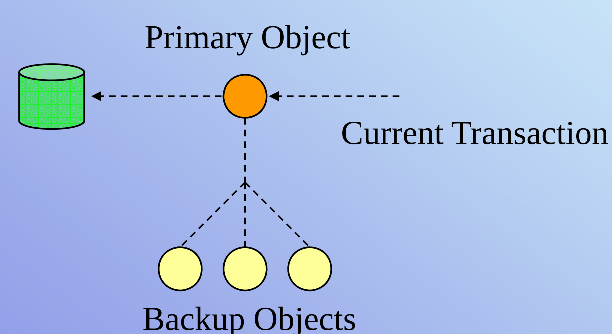
Replication Style Support (cont.)

- Warm-Passive Replication:
  - In this case, the primary object should update the backup objects in the scope of a RAT.



Replication Style Support (cont.)

- Cold-Passive Replication:
  - Checkpointing should be performed in the scope of the transaction that the primary object participates in it.



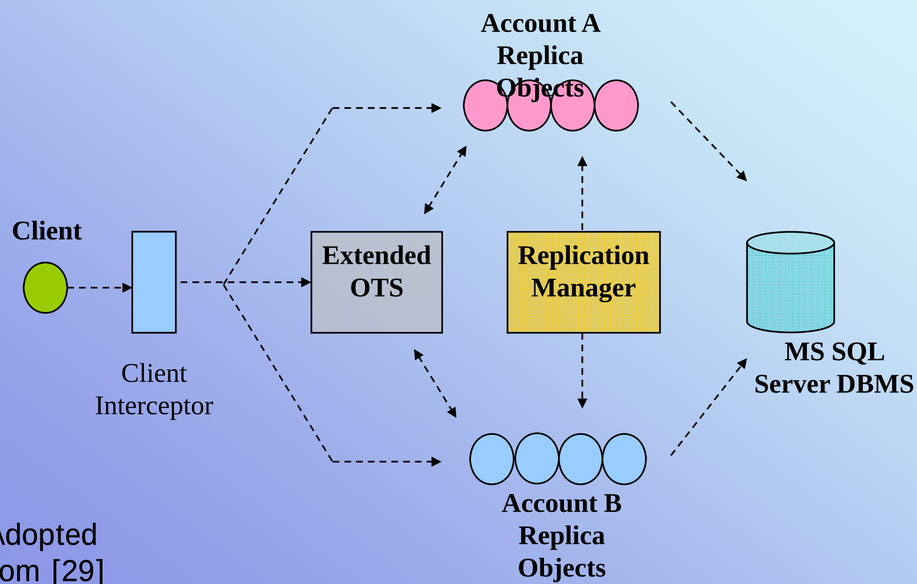
Implementation

- My Implementations:
  - Extending OTS in such a way that can support roll-over approach.
  - Implementing a light-weight Replication Manager.
  - Implementing a prototype to evaluate our model.

Replication Manager

- Our implemented replication manager:
  - Keeps an object factory for each host
  - Recreates objects whenever they fail
  - Keeps object groups as a bunch of replica object
  - Assigns each object group a set of properties (e.g. minimum number of replica objects)

The Prototype



Adopted from [29]

Measurement Parameters

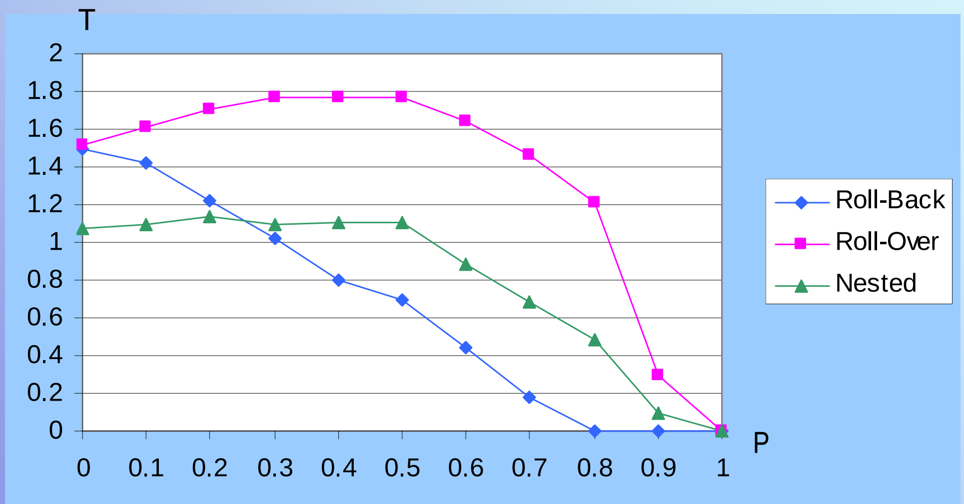
- We ran the implemented prototype with the change on:
  - Number of Replica Objects (N)
  - Failure Probability of a replica object (P)
  - Transaction Model (TM)
- And for each run we measured:
  - The overall transaction throughput (T) in terms of the number of committed transaction per second.

Performance Evaluation

- We compared the transaction throughput of our approach with the transaction throughput of:
  - Felber's approach [20], which uses nested transactions
  - Zhao's approach [29], that uses roll-back approach.

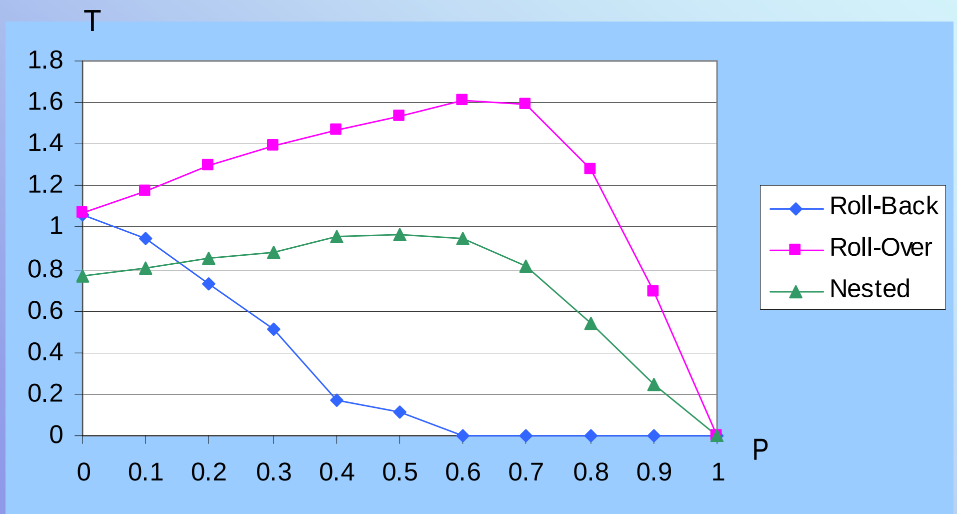
Experimental Results

- N=2



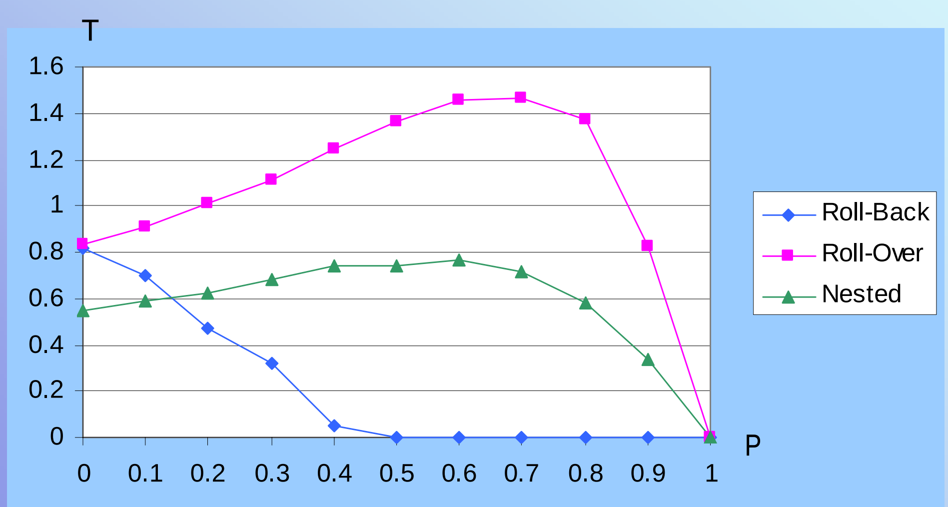
Experimental Results (cont.)

- N=3



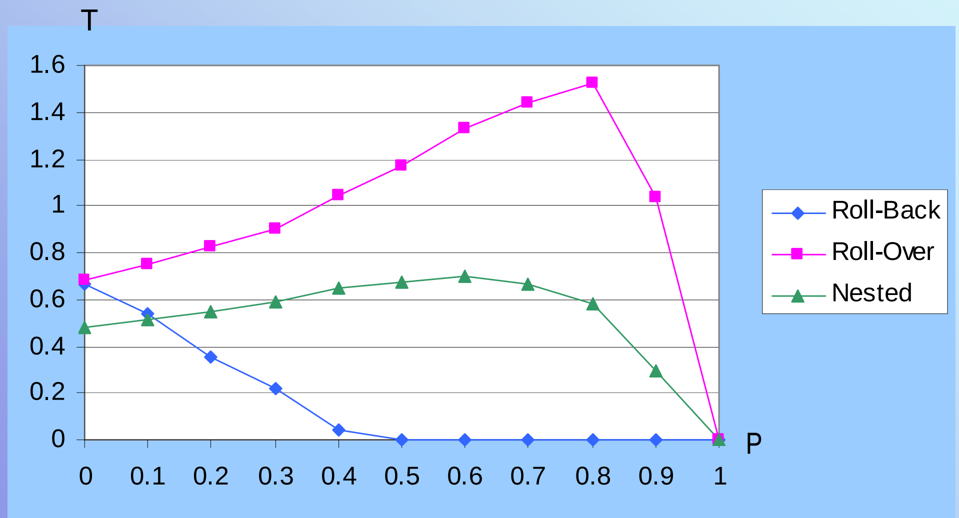
Experimental Results

- N=4



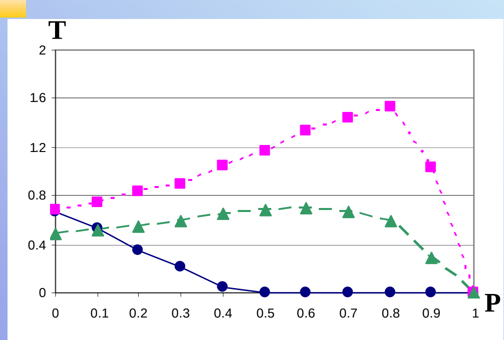
Experimental Results

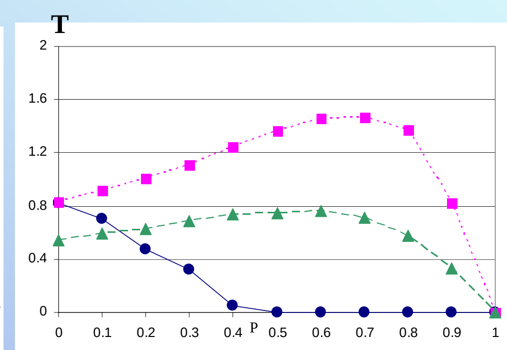
- N=5



All together

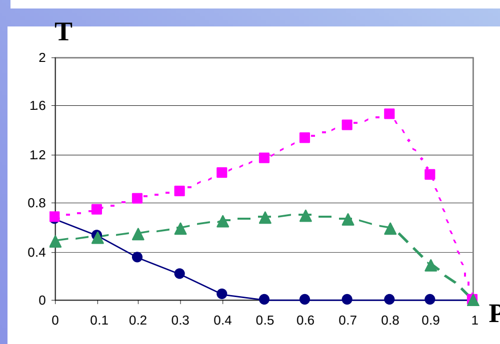
N=3

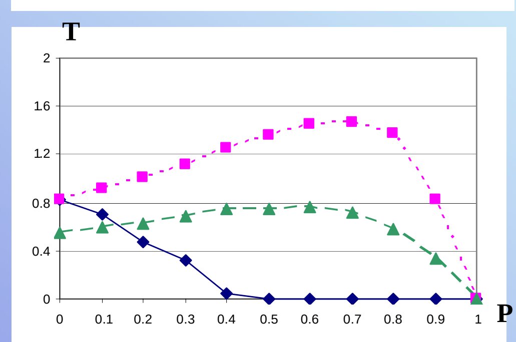




N=2

N=5

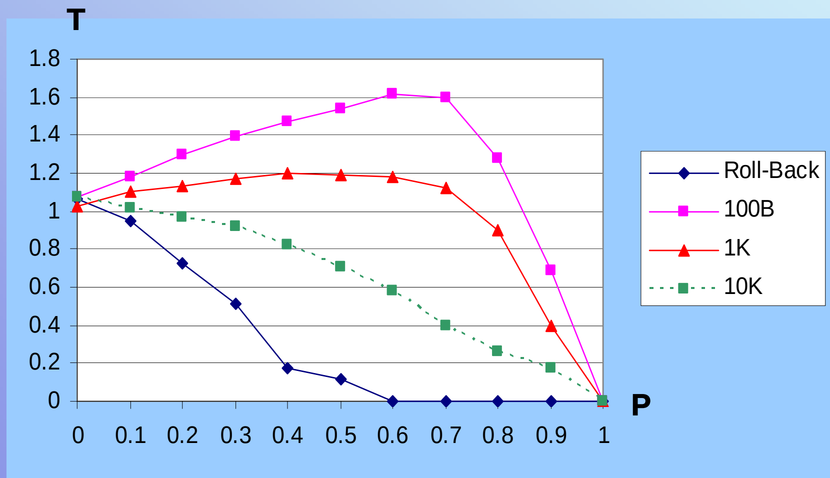




N=4

Object Size

- The object size can affect the performance of RATs, because the object state needs to be transmitted on each object recreation.



Conclusion

- We showed that current replica consistency techniques that are based on total order multicast protocols cannot guarantee system safety.
- We presented a new transaction model that can be applied to replicated objects.
- In this model, a failure in the scope of a transaction that is running on a group of replicated objects can be ignored.

Conclusion (cont.)

- We implemented a prototype to evaluate this extension.
- The experimental results show that this model is particularly beneficial:
  - When dealing with crowded object groups
  - In faulty environments
  - For light weight objects

Other Open Issues

- Active_With_Voting (Quorum-Based or Consensus-Based) Replication [33]
- Applying transactions on active with voting replication style
- Fault Tolerant CCM
- Modeling of this approach [32]
- Transaction and Concurrency Integration

References

[1] Object Management Group, "The Common Object Request Broker: Architecture and Specification, 2.6 Edition," OMG Technical Committee Document formal/02-01-02, Jan. 2002.

[2] D. Slama. J. Garbis, and Russell P. "*Enterprise CORBA*", Prentice Hall, 1999

[3] J. C. Laprie, "Dependable Computing: Concepts, Limits, Challenges," *Proceedings of PTCS-25,* Pasadena, 1995, pp. 42-54.

[4] G. Weikum, G. Vossen, "Transactional Information Systems", Academic Press, 2002, San Francisco, CA, USA.

[5] P. Narasimhan, "*Transparent Fault Tolerant for CORBA*", PHD thesis, University of California, December 1999.

[6] Object Oriented Concepts, Inc., "CORBA/C++ Programming with ORBacus", Student Workbook, Version 1.0.5, December 2000. http://www.ooc.com

[7] L. Pullum, "Software Fault Tolerance Techniques and Implementations", Artech House, ISBN 1-58053-137-7

[8] A. S. Tanenbaum, M. V. Steen, "Distributed Systems: Principles and Paradigms", Prentice Hall, 2002, ISBN: 0-13-088893-1

[9] Guerraoui, R. and Schiper, A. (1997) Software-Based Replication for Fault Tolerance. *IEEE Computer*, 30(4) pp. 68-74.

References (cont.)

[10] Object Management Group, "Object Transaction Service Specification", Version 1.4, OMG Technical Committee Document, formal/03-09-02, September 2003.

[11] Object Management Group, "Fault Tolerant CORBA (Final Adopted Specification)", OMG Technical Committee Document formal/01-12-29, Dec. 2001.

[12] Felber, P., Guerraoui, R., and Schiper, A. (2000) Replication of CORBA Objects. *Lecture Notes in Computer Science,* 1752, Springer Verlag, Berlin, pp. 254-76.

[13] D. Powell, "Distributed Fault Tolerance: Lessons form Delta-4", IEEE Computer, 1994.

[14] G. D. Parrington, S. K Shrivastava, S. M. Wheater, M. C. Little, "Design and Implementation of Arjuna", Technical Report, Department of Computing Science, University of Newcastle,

[15] IONA and Isis, An Introduction to Orbix+Isis, IONA Technologies Ltd. and Isis Distributed Systems, Inc., 1994.

[16] S. Matfeis. "*Run-Time Support for Object-Oriented Distributed Programming*". PhD thesis, University of Zurich, February 1995.

[17] B. Natarajan, A. Gokhale and S. Yajnik, "DOORS: Toward High-performance Fault Tolerant CORBA", in the proceedings of the 2nd Distributed Objects and Applications (DOA) Conference. Antwerp, Belgium, September 2000.

[18]

[19] L. E. Moser, P. M. Melliar-Smith, and P. Narasimhan, "Consistent object replication in the Eternal system", *Theory and Practice of Object Systems,* 4(2):81-92, 1998.

References (cont.)

[20] P. Felber, P. Narasimhan, "Reconciling Replication and Transactions for the End-to-End Reliability of CORBA Applications", in the proceedings of international symposium on Distributed Objects and Applications (DOA'02), pp. 737-754, October 2002.

[21] D. R. Cheriton and D. Skeen, "Understanding the limitations of causally and totally ordered communication", Proc. of 14th ACM Symp. on Operating Systems Principles, Operating Sytems Review, 27 (5), pp. 44-57, December 1993.

[22] Rebuttals from Cornell, Operating Systems Review, 28 (1), January 1994.

[23] A. Schiper and M. Raynal, "From Group Communication to Transactions in Distributed Systems", CACM, 39(4), April 1996.

[24] M. Martynezy, R. Peris, and S. Evalo, "Group Transactions: An Integrated Approach to Transaction and Group Communication", Workshop on Concurrency in Dependable Computing, June 2001.

[25] S. Frolund and R. Guerraoui, "Implementing e-Transactions with Asynchronous Replication," IEEE Transactions on Parallel and Distributed Systems, vol. 12, No. 2, pp. 133-146 (2001).

[26] M. C. Little and S. K. Shrivastava, "Integrating Group Communication with Transactions for Implementing Persistent Replicated Objects," Lecture Notes in Computer Science, Vol. 1752, Springer-Verlag, 2000.

[27] Felber P. and Narasimhan P. "Experiences, Strategies, and Challenges in Building Fault-Tolerant CORBA Systems", IEEE Transactions on Computers, Vol. 53, No. 5, May 2004

References (cont.)

[28] S. Frolund and R. Guerraoui, "CORBA Fault Tolerance: why it does not add up?" , In Proc. of the IEEE Workshop on Future Trends in Distributed Systems, Cape Town, December 1999.

[29] W. Zhao, L. E. Moser and P. M. Melliar-Smith, "Unification of Replication and Transaction Processing in Three-Tier Architectures", In Proc. of the International Conference on Distributed Systems, 2002.

[30] M. Cukier, J. Ren, C. Sabnis, W.H. Sanders, D.E. Bakken, M.E. Berman, D.A. Karr, and R. Schantz, "*AQuA: An Adaptive Architecture that Provides Dependable Distributed Objects,*" Proc. IEEE 17th Symposium on Reliable Distributed Systems, pp. 245-253, Oct. 1998.

[31] S. Matfeis. "*Run-Time Support for Object-Oriented Distributed Programming*" PhD thesis, University of Zurich, February 1995.

[32] I. Majzik and G. Huszerl, "*Towards Dependability Modeling of FT-CORBA Architectures*", In Proc. 4th European Dependable Computing Conference (EDCC-4), pp. 121-139., Toulouse, France, 23-25 October 2002.

[33] J. C. Martinez and J. F. Paris, "*A New Voting Approach to Fault-Tolerant CORBA*", Proceedings of the International Conference on Software Engineering Applied to Networking and Parallel/Distributed Computing (SNDP '00), Reims, France, May 2000, pages 358-365.

Thanks ...

Hadi Salimi

h_salimi@mail.iust.ac.ir

January 14, 2005