

Network Design for the DSM Operating System Plurix

M.Wende, M. Schoettner, O. Schirpf and P. Schulthess
University of Ulm
Germany

June 4, 1999

Abstract

For the cluster operating system Plurix a network protocol was designed and implemented, which provides special support for fast network memory access and consistency. Plurix is a lean Java OS basing on the well known concept of distributed shared memory (DSM). Distributed memory management simplifies higher OS layers but imposes special requirements for memory consistency and networking - especially for the design of network device drivers and network protocols.

1 Introduction

DSM consistency is guaranteed by the novel Plurix transaction protocol. Transactions are well known from distributed databases, where parallel access to common data is possible. Typically locking mechanisms are used to serialise concurrent data access. Kung and Robinson [3] presented the method of optimistic synchronisation instead of such locking mechanisms. Using optimistic synchronisation all operations are treated simultaneously. In case of colliding operations all except one of the operations in question are rolled back and started again.

In this paper we present a network protocol that combines standard protocols like UDP / IP and functions, needed for transactions in the operating system Plurix. Both, the protocol and special driver functions are needed because the network layer and transactions are responsible for the memory consistency in the DSM. This paper is divided into five sections. Following this introductory section, section two presents the transaction architecture of Plurix. Section three discusses the benefits and handicaps of standard network hardware in the context of transactions. In section four we introduce the transaction network protocol and finally we describe the current implementation status and illustrate future work areas.

2 The Plurix transaction architecture

Objects in a distributed environment can be accessed simultaneously by different users. Unprotected concurrent access on objects may result in inconsistent multiple versions of the same data in the cluster. Plurix allows concurrent access during operations but at the end of a set of operations only one write operation to an object becomes visible - a transaction. This concept known as optimistic synchronisation or optimistic concurrency

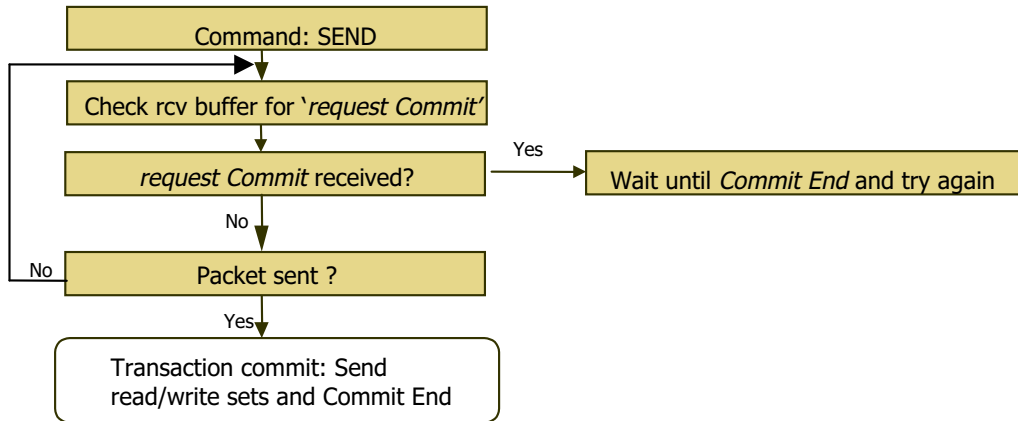


Figure 1: The two phase commit message protocol

control (OCC) [3] guarantees strict memory consistency at the end of each transaction. This is achieved by means of the memory management unit (MMU) and a transaction strategy. The MMU monitors all read and write operations on memory pages from the beginning of a transaction (BOT) until the end (EOT). At EOT the addresses of the memory pages which have been read or written - the read write sets - are currently broadcasted to the whole Plurix cluster, read/write collisions between transactions are detected and transactions are committed or aborted depending on the transaction strategy used. Plurix uses a 'first-wins strategy' for committing / aborting of transactions. The first transaction, which sends the read/write set wins (commits) - all others colliding with this one are aborted and restarted. During a transaction undo or redo sets are saved, allowing resetting and restarting.

3 The network hardware

The first-wins strategy requires all Plurix stations to identify the same message as winner between transactions. Therefore a global unique packet order is needed. Using a network with a bus topology, like 10MBit/s standard, non switched Ethernet, guarantees a global packet order for all members except for the sender. All receivers get the packets in the network order in their receive buffer, only the sender does not receive his own packet. The sender moves the data into the send buffer of the NIC and commands the NIC to send the packet. Due to the CSMA/CD protocol the packet often can not be sent immediately, because the NIC has to wait for a free slot on the net.

4 The network protocol

The Plurix network protocol DSM-P has to solve two problems:

- Determine identical packet order for all Plurix stations,
- Prevention of overlapping commit messages.

During the commit phase of one transaction all others have to wait until the commit message is complete. Parallel committing can cause deadlocks between different transactions. For example, the read/write sets of two transactions containing the same address could delete the object with the common address cross-wise: the object will be lost.

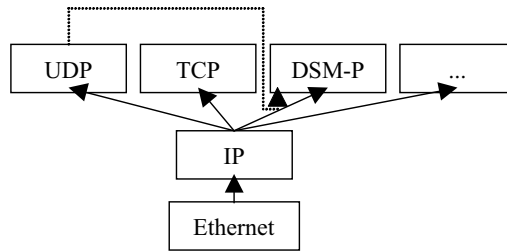


Figure 2: The Plurix network protocol architecture

4.1 Two phase commit messages (2PCM)

DSM-P uses two phase commit messages (2PCM) in order to guarantee object consistency.

1. Prepare for commit: The Plurix-Cluster starts the commit phase. Every transaction knows that one of the next received packets must be the second phase
2. Commit: now the read/write sets are sent by broadcast messages. This message also contains the Commit-End signal that allows other transactions to send their prepare for commit message.

The senders packet order is also guaranteed using this protocol: Between the prepare for commit message and the send interrupt the sender checks the receive buffer for received prepare for commit messages of other participants. If there is no message in the receive buffer the sender is the first on the net.

If the sender receives between the send command and the "packet sent" interrupt a prepare for commit message of an other Plurix computer he has to wait for committing until the others commit has finished. Figure 1 illustrates this from the point in time when the request for commit message is in the send buffer of the NIC. The receive buffer is checked for an incoming request commit message until the own message is sent successfully. If - during this time - a request commit message was received the own message has to wait until the reception of a commit end.

5 Further Work

The protocol presented is stable and fast on raw Ethernet. Currently a new protocol hierarchy will be implemented, that combines DSM-P with the standard protocols IP/UDP including ARP and IGMP (Figure 2).

In future there should be the possibility to start Plurix in a Browser environment. This is one of the current objects of our research. The new implemented network protocol will support such an environment by a special version where DSM-P uses UDP as underlying protocol. Then, the "Plurix-Applet" user can communicate with other Plurix nodes via the internet through gateways. When DSM-P only uses IP, users can also participate in a remote Plurix cluster over gateways. For both a special transaction protocol, have to be developed that enhances or replaces the first-wins strategy used today.

6 Conclusion

This paper presented a network protocol specialised for transaction processing within Plurix. The combination of DSM, transactions and efficient networking leads to an ele-

gant and simple operating system architecture. Collisions between competing transactions may be resolved using different strategies. A simple strategy is the "first-wins" strategy. The commit message of a winning transaction always causes the roll-back of all other transactions and eliminates all inconsistent object duplicates without additional communication. For this reason the aborted transactions can be restarted immediately.

Based on the available components and synergies more complex DSM management strategies will be studied in future projects.

References

- [1] M. C. Carey. Improving the performance of an optimistic concurrency control algorithm through timestamps and versions. *IEEE Transactions on Software Engineering (TSE)*, 13(6):746–760, 1987.
- [2] Maurice Herlihy. Apologizing versus asking permission: optimistic concurrency control for abstract data types. *ACM Transactions on Database Systems*, 15(1):96–124, 1990.
- [3] H.T. Kung and Robinson John.T. On optimistic methods for concurrency control. *ACM Transactions on Database Systems*, 6(2):213–226, 1981.
- [4] Peter Schulthess et al. Dsm-java: Foundation of a lean distributed operating system. *Proceedings of the International Workshop on Distributed Computing on the Web, Rostock, Germany, 1999*.
- [5] Alexander Thomasian. Distributed optimistic concurrency control methods for high-performance transaction processing. *IEEE Transactions on Knowledge and Data Engineering*, 10(1):173–188, 1998.
- [6] Stefan Traub. *Speicherverwaltung und Kollisionsbehandlung in transaktionsbasierten verteilten Betriebssystemen*. PhD thesis, University of Ulm Germany, 1996.