Monitoring and Managing Parallel Communication Architectures

Dipl.-Inform. T. Braun, Dipl.-Inform. C. Schmidt, Dipl.-Inform. J. Seitz
University of Karlsruhe
Institute of Telematics
P.O. Box 6980
D-7500 Karlsruhe
Federal Republic of Germany
braun@telematik.informatik.uni-karlsruhe.de
schmidt@telematik.informatik.uni-karlsruhe.de
seitz@telematik.informatik.uni-karlsruhe.de

Abstract

The administration of communication networks becomes a more and more important factor. It is not tolerable if a communication system works with a reduced quality or even is not available. On the other hand, the growing need for high speed communication does not allow to add additional overhead for management functions to these communication systems. This paper presents an architecture which allows an efficient management while reducing the processing overhead in the network nodes caused by management functions simultaneously. This goal is achieved by placing the management entity on a completely autonomous working device and therefore, a parallel processing of management and communication functions is possible.

The management is achieved by the Simple Network Management Protocol version 2*, the successor of the widely accepted standard protocol for internet management, SNMP. This papers describes the integration of SNMP-2 and NETMON (NETwork MONitor), a hybrid monitoring tool developed at our institute. NETMON – originally developed for monitoring distributed systems – has been extended to supervise parallel communication systems.*

1 Introduction

Due to the progress in fiber-optic and VLSI technologies, networks offering increasing transmission capacity and decreasing error rates are becoming available. New applications would benefit from this bandwidth, but communication systems have not kept up with the available raw transmission rates. One concept to speed up communication systems is a parallelization of several protocol functions. This paper deals with monitoring and managing of such a parallel protocol implementation architecture. It is obvious, that the processing overhead imposed by management functions should be reduced to a minimum. Therefore, we introduce a management entity which is located on a completely autonomous device combined with a monitoring tool, which is designed for the monitoring of high speed protocol implementations.

The monitoring is realized by NETMON, a monitoring tool developed at our institute. Section 2 gives an overview of different kinds of monitoring approaches and introduces basic concepts of NETMON, which are relevant to this paper. Section 3 briefly describes the Simple Network Management Protocol version 2 and the improvements of version 1. In section 4 we combine both concepts in the architecture of an innovative management entity, which is especially designed to monitor and manage parallel communication protocol implementations. Section 5 illustrates this architecture with two examples and, finally, section 6 gives a summary of this paper and shows the current progress.

2 Monitoring Parallel Implementation Architectures for Communication Protocols with NETMON

To satisfy the growing need for high speed communication one solution is to separate different parts of a communication protocol in order to execute these parts in parallel. Common approaches are implementations of standard protocols such as ISO or Internet protocols. Because these protocols often restrict the implementation on parallel architectures, advanced protocol architectures based on parallel concepts are currently being developed [BZ92]. The use of multiprocessor architectures like transputer networks as a platform for high-performance protocol implementations is a promising approach, which is investigated by a considerable number of research projects [Zit91].

For optimizations and to achieve highly efficient parallel protocol implementations, a detailed performance analysis of the real implementation is required. Due to efficiency reasons, the loads of several processors of a multiprocessor architecture have to be balanced and idle processor periods have to be avoided. The detailed analysis requires powerful monitoring tools, which must provide high resolution, permit only low interferences among the monitoring system and the monitored object, and support a global view of the states of all the processors in a multiprocessor architecture.

In order to monitor a communication system there are three realizations possible [RS92]: (1) hardware-monitoring, (2) software-monitoring and (3) a hybrid solution.

A *hardware-monitor* is a stand-alone system with its own network attachment. It observes the signals on the physical level and analyses them. It can even initiate traffic when special measurements are needed. Although, this kind of monitors do not influence the system to be monitored they have some important disadvantages. On the one hand they must gain the desired information out of physical signals. This prohibits concrete results about the higher layers of the protocol stack. On the other hand they are too expensive to supply a monitor for each network node.

The second solution – the software-monitors – depend on procedure calls that have to be integrated into the monitored software. Therefore, the monitoring process uses CPU and memory and, thus, inhibits the monitored software to some degree. But on the other side software-monitors are more flexible and cheaper than hardware-monitors.

The third kind of monitoring is the so called hybrid approach. The monitored software is sending out some kind of events and a dedicated hardware entity is analyzing these events partly supported by the hardware on which the monitored software is running. This solution is a good compromise between the hardware- and the software-monitors combining the advantages of both and minimizing their disadvantages. Therefore, we have chosen the hybrid solution: the monitoring system NETMON (cf. figure 1). NETMON has initially been designed for monitoring distributed systems, but has been extended for the use in multiprocessor environments such as transputer networks.

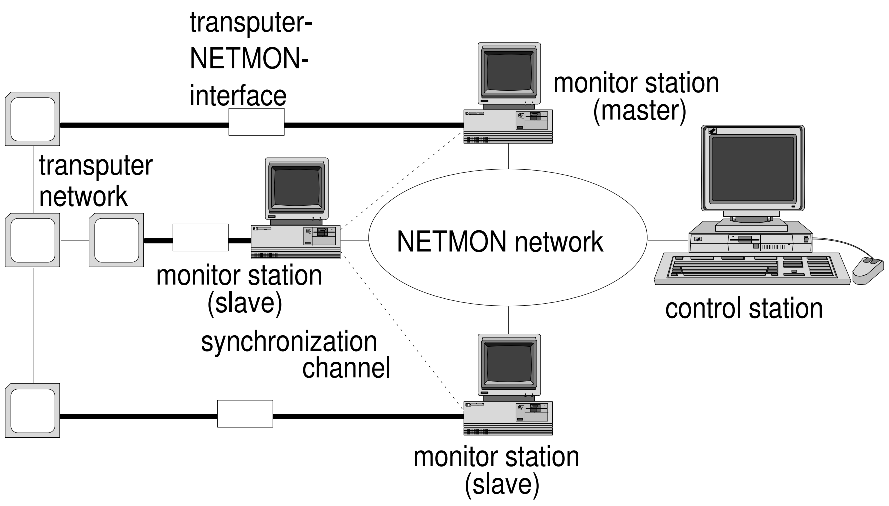

Figure 1: The NETMON architecture

NETMON consists of one central control station and several monitor stations, which are associated with the monitored objects. The monitor stations react on events from the monitored objects. Usually, simple software operations with very low overhead (< 1 microsecond on a transputer), which have to be inserted into the monitored software, generate these events. The event values are recorded and a time stamp is added to each event value by a monitor station. Each monitor station collects the events to an event stream. All event streams are sent over an interconnecting network to the central control station, which provides tools for on-line and off-line evaluation.

NETINT (NETMON Interpretation Tool) has been developed for off-line evaluations of NETMON measurements. It allows a more detailed and complex analysis of measurement data and offers better display facilities of the results. As input NETINT expects the measurement data recorded during a measurement experiment and a description file for their interpretation. For evaluation description we developed a special description language and a corresponding interpreter [Gla92]. The pre-processor NETINTPP substitutes strings in a description file as defined in a substitution file.

The added time stamps are generated by the local timers of the monitor stations. To achieve a global and consistent view to all event streams, the local timers have to be synchronized. Synchronization is initiated by a dedicated monitor station, which acts as a synchronization master and sends synchronization signals to the slave monitor systems, which have to synchronize their local timers. NETMON achieves resolutions of 125 nanoseconds.

NETMON is connected by the transputer-NETMON-interface to the data/address bus of a transputer. The interface records each writing access to a memory address (NETMON address) with A31 = 0 and A30 = 0, takes the written value and delivers it via a 8 bit wide Centronics-like interface to a NETMON monitor station, which finally creates the complete event record. The events to include in the software must have the form

{int *netmon_address = 0; *netmon_address = value}.

To get a detailed view of the behavior of the processes running at a transputer network, it is necessary to insert events in all the processes. On the one hand in order to gain information about the performance of a parallel protocol architecture, events have to be inserted before and after operations, which may cause a context switch, e.g., communication operations for exchanging messages among transputer processes. The insertion of events can be performed automatically by a pre-processor before compilation of the transputer process code. On the other hand for management purpose events have to be associated with certain actions within the protocol that change the state of Managed Objects, e.g., whenever the protocol must retransmit a packet an event is generated to update the counter for retransmitted packets.

3 Overview on SNMP-2

The importance of network management has triggered many standards committees to come up with network management models and protocols. The standard to become widely accepted is the *Simple Network Management Protocol (SNMP)*, as it has been developed for the internet [CFSD90]. As a follow-on to this protocol the new *Simple Network Management Protocol Version 2 (SNMP-2)*[1] has been introduced in 1992 to overcome some shortcomings of SNMP. Nevertheless, the main ideas of SNMP shall be pursued still: "minimizing the cost and complexity of network management Agents and putting net management in centralized stations" [Jan92]. The services and the underlying network model of SNMP-2 shall be detailed in this section.

SNMP-2 relies on a management model consisting of so called *Managed Objects*. The structure of these objects is defined within the *Structure of Management Information (SMI)* [MR88b, RM90, RM91]. The definition of each Managed Object therefore consists of roundabout eight parts: the object identifier, its syntax, the access mode (read-only, read-write, write-only or not-accessible), the status (mandatory, optional, obsolete or deprecated), the description, the reference, the indexing part and the definition of the default value [RM91]. Some of these parts are optional since they may not make sense with every Managed Object, e.g., the Managed Object to describe the network node's system (sysDescr) does not need indexing or a default value. The definition of a Managed Object is done by using a so called Object-Type Macro defined in *Abstract Syntax Notation 1 (ASN.1)* [Bau93]. For the type definition of Managed Objects there are primitive types (*Integer*, *Octet String*, *Object Identifier* or *NULL*), constructor types (by using the ASN.1 *sequence*–operator) and defined types like *NetworkAddress*, *IpAddress*, *Counter* (an integer that can only be increased), *Gauge* (an integer that can be increased and decreased), *Time Ticks* (measuring time in 1/100 seconds) or *Opaque*.

The managed objects are gathered in a *Management Information Base (MIB)*. In order to define a basis for network management

[1] In [Jan92] SNMP-2 is also titled *Simple Management Protocol (SMP)*.

two MIBs have already been standardized, namely MIB-I [MR88a] and MIB-II [MR91]. To address a Managed Object in a MIB there exists a naming tree, in which all nodes are identified by a name and a number. Considering the before mentioned Managed Object sysDescr, this object can be found in MIB-I on the route consisting of the nodes iso (1), org (3), dod (6), internet (1), mgmt (2), mib (1), system (1) and sysDescr (1). It therefore can be addressed by concatenating the node identifiers: 1.3.6.1.2.1.1.1. To distinguish between the (generic) Managed Object and the Managed Object's instance the latter one will have an additional identifier, a so called variable name. For simple Managed Objects like sysDescr the instance will be identified by adding a "0" to the Managed Object's name (→ 1.3.6.1.2.1.1.1.0). For Managed Objects representing tables the variable name will be an index to a special row (therefore, you need the index part).

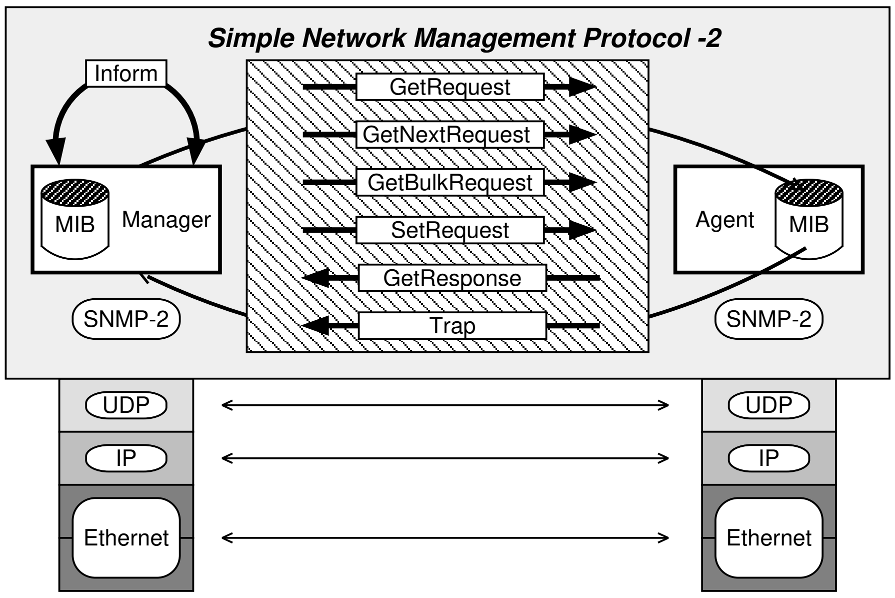

Figure 2: Management via SNMP-2

For the management task SNMP-2 provides two different types of processes: the *Manager* and the *Agent*. The first one is the means for a management application to access the MIBs, the second one is responsible for the administration of one MIB. For the exchange of information between two SNMP-processes SNMP-2 offers six services:

1. *Get*: This service is used to get the contents of a Managed Object stored in the MIB of a network node.

2. *GetNext*: By this service it is possible to easily browse through the MIB of a network node, because the delivered value belongs to the Managed Object that is the lexical successor to the given Managed Object.

3. *GetBulk*: In order to minimize the amount of *Get/GetNext* requests when accessing a tabular Managed Object (e.g., a routing table) this service has been defined for SNMP-2.

4. *Set*: This service provides a means to change the value of a Managed Object.

5. *Trap*: This service – invoked by an Agent – is used to notify a Manager of an exceptional situation.

6. *Inform*: This service is used to allow communication between Managers. One should notice that this service is not supplied by SNMP version 1.

A further development of SNMP-2 is the realization of security, that makes it hard (if not impossible) for applications faking to be an authorized management process. Thus, the full power of the *Set*–service can be opened to the designers of management applications, which the vendors of manageable equipment have not dared to allow with SNMP version one.

Another shortcoming of SNMP has been removed: SNMP-2 is not deeply interconnected with TCP/IP but may also be mapped onto protocol stacks like IPX, Appletalk or even OSI's connectionless network protocol (CLNP). This is another step away for SNMP being only a short term solution, which will be replaced by OSI management once its practicability will have been proved [CFSD90].

Figure 2 illustrates a scenario for the management via SNMP-2.

4 Architecture

In this part we introduce the architecture of a management entity for parallel high performance communication protocols. The design goals are to minimize the processing overhead imposed on the communication system and to achieve an efficient and simple management simultaneously. To support precise management of parallel high performance communication protocols a high resolution and a low interference with the managed system are necessary.

The management architecture includes the functions to monitor and control communication protocols. It consists of four main components: NETMON, the local management facility, the SNMP (Simple Network Management Protocol) facility, and the Management Information Base (MIB). This modular design allows a separation of monitoring, local control facilities and global control facilities. All monitoring functions are located in NETMON, the local facility administrates the MIB and the SNMP facility performs the communication with the Manager using SNMP. Figure 3 depicts the management architecture, and the communication between its components.

NETMON represents the interface to the communication system. Required network management information can be derived from events generated by the communication software in the network node. With NETMON there are no 'black box' measurements possible, because the source code has to be modified. But to incorporate events in the monitored software, there are only simple software operations necessary, which produce low overhead. All generated events are received by NETMON, where a selector utility enables filtering of events. To reduce the amount of measured data to informations which are important for management functions, the selector allows the definition of events that have to be sent to the local facility and those that can be discarded. Thus, the sources may simultaneously include events for performance measurements, which are filtered by NETMON as well as events for management purposes, e.g., a counter for retransmitted PDUs.

The *local facility* receives management information from NETMON, updates the MIB, and decides whether the new information requires modifications of local parameters or a notification of the Manager. The examples in section 5 illustrate the different cases. Each event is marked with a time stamp and moreover,

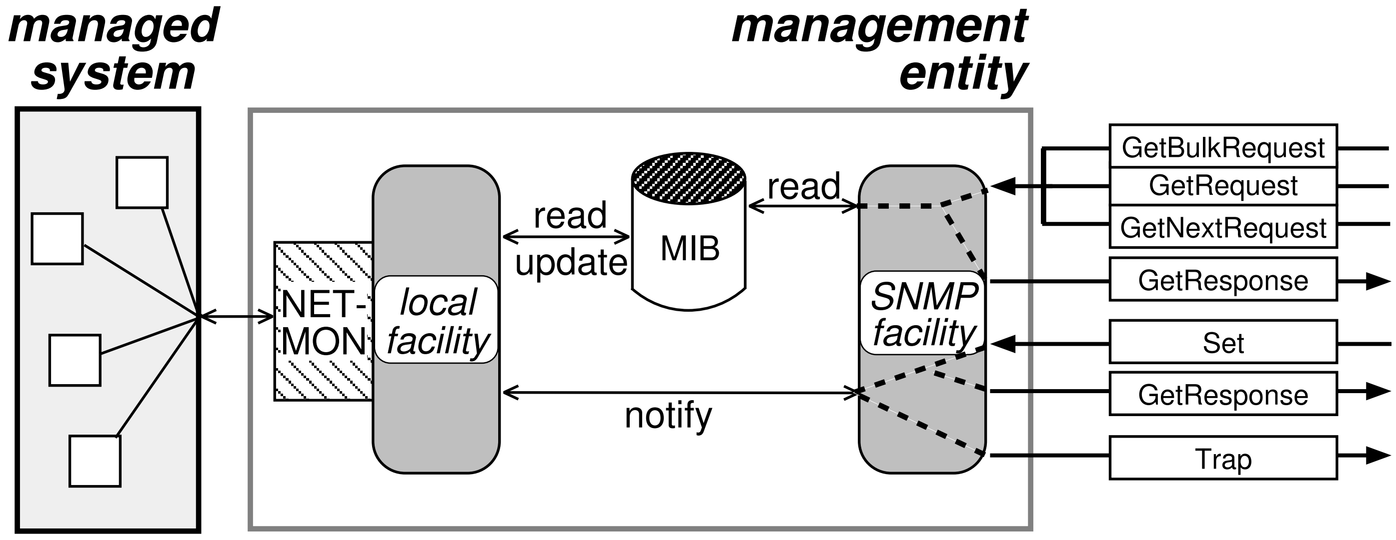

Figure 3: The Management Architecture

events can be combined to event pairs or streams of events by defining predecessors and successors. With these functions it is possible to realize MIB elements like strings (e.g., names of objects), counters, time intervals (by pairs of events), addresses and gauges. The local facility collects all events, computes the corresponding elements of the MIB, and updates the MIB. Simple MIB elements can be computed from one event (e.g., counters, strings), but complex elements (e.g., time intervals) may need two or even more events for their computation. Updating the MIB is not a difficult or time-consuming task, but MIB elements may be coupled with a *threshold attribute*. Each time the element is updated, the actual value must be compared with a threshold value, and if the actual value crosses the threshold the Manager must be notified. In this case the local facility informs the SNMP facility, which is responsible for creating the corresponding SNMP *Trap* message.

Threshold-attributes are not restricted to a single value, but may also be a set of values, where each value represents a different level of severity. A notification of the SNMP facility is triggered each time the MIB element reaches any of these values. Furthermore, it is possible to distinguish between *counter-thresholds* and *gauge-thresholds*. In the case of a counter-threshold a notification is triggered if the actual value crosses the threshold, gauge-thresholds may be crossed in two directions, e.g., a counter for retransmissions may cross its counter-threshold, but the number of retransmissions in a specific period might have been higher than its gauge-threshold in the last period, while the value is lower in the actual period. To avoid repeated notifications when the value makes small oscillations around its gauge-threshold, an additional *hysteresis mechanism* is provided. This means, if a notification was once triggered, no further crossings will trigger a notification unless the actual value leaves the short interval between threshold and hysteresis value. To notify the Manager about values which have crossed their thresholds we introduce new *Traps* to the already existing 7 SNMP Traps. For each counter-threshold we need one additional trap, for each gauge-threshold there must be two additional traps, one for each direction.

As can be seen, locating the management function on a autonomous working device releases the communication software from time consuming management functions, which is an important issue especially for high performance protocols.

The *MIB* in our architecture consists of the MIB-II as well as a private part with variables needed for the parallel communication protocols. We have chosen a redundant placement of a part of the MIB on the managed network node to reduce the communication overhead between the managed network node and the management entity. The part of the MIB stored in the managed node includes all static information and also information which needs no frequent adaptation. All other actual information is only stored in the MIB on the management entity. Periodically, it is necessary to update the part of the MIB on the managed network node and therefore, there exists a special process on this node, which receives the new information from NETMON and updates the MIB. One example for information which does not change frequently is a routing table. The managed network node needs the routing table in order to function properly. Whenever this table must be updated, the SNMP facility informs the local facility of this change. The local facility updates the Managed Object modelling this routing table and propagates the change to the managed system.

The *SNMP facility* performs the communication with the Manager or SNMP facilities in other hosts. This communication is based on the Simple Network Management Protocol. The SNMP facility acts in two different ways. As an Agent process it informs the Manager about critical situations and sends responses to incoming requests. If the management entity itself needs information from other hosts, the SNMP facility operates as a Manager by sending requests to other hosts. This change between two operation modes allows a central as well as a distributed decentralized management.

Working as Agent the SNMP facility receives *Get* or *GetNext* messages from the Manager. In this case it performs a read operation on the MIB, creates a *GetResponse* and sends the retrieved information back to the Manager. Receiving a *Set* message, the SNMP facility informs the local facility and afterwards sends a *GetResponse* to the Manager. To avoid access conflicts, updates of MIB values are only performed by the local facility. Furthermore, the SNMP facility receives messages from the local facility whenever the Manager should be informed about changes in the managed node. The SNMP facility creates a *Trap* message and sends it to the Manager.

If the management entity operates as management station, the SNMP facility sends *Get* or *GetNext* messages to other network nodes in order to retrieve new information. But it is not allowed to use the *Set* command to change the MIB of other network nodes.

The two management components (local and SNMP facility) in a single host cooperate over the MIB and exchange messages to inform the counterpart about important changes. All SNMP concerning issues are separated from local issues and are located in the SNMP facility. This modular design has many advantages, especially, it allows a simple integration of future changes.

5 Two Examples

The way the described management entity is working is now illustrated by two examples. The first one shows how the local management facility can improve the local behavior. Let there be a gauge for the number of retransmitted PDUs over time. This gauge can be associated with a lower and an upper threshold (see figure 4). Whenever the lower threshold is reached, the local facility tries to improve the behavior by modifying local parameters. Therefore, it must take the number of negatively acknowledged PDUs into account. If this number (stored in a Managed Object as well) is way beneath a critical value, then there will be a high possibility of the time value for the retransmission timer being set too low. By increasing this parameter on the local parallel communication architecture, the number of retransmitted PDUs might decrease. If it does not or if it even reaches the higher threshold, the local facility must trigger the SNMP facility, which sends out a trap to inform the Manager.

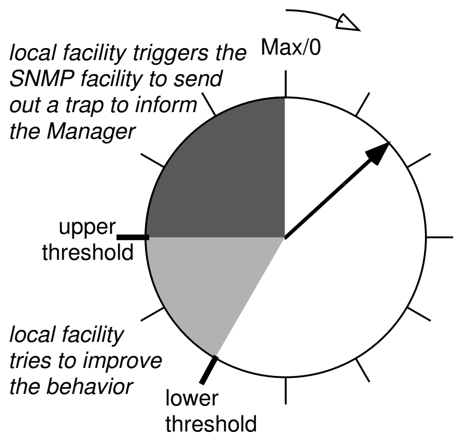

Figure 4: Gauge for the Number of retransmitted PDUs over time

The second example considers the possibility of the management entity acting as a Manager. Let there be a gauge for packets received with a checksum error within a given period in each station as depicted in figure 5. This gauge can be associated with a threshold as in the example before. But unlike before the SNMP-facility must react on the gauge crossing the threshold. In the given example the local facility of node A notices that the threshold for PDUs with checksum errors is crossed. It informs the SNMP facility, that tries to get information about checksum errors from Agents that are directly connected (stations B and C). Station B's SNMP facility replies with a number of checksum errors that has just reached the threshold. Thus, station A's SNMP facility might conclude that the line between node A and node B is probably erroneous. In this case it informs the concerned Managers about that line and may even initiate tests for the transmission line under suspicion.

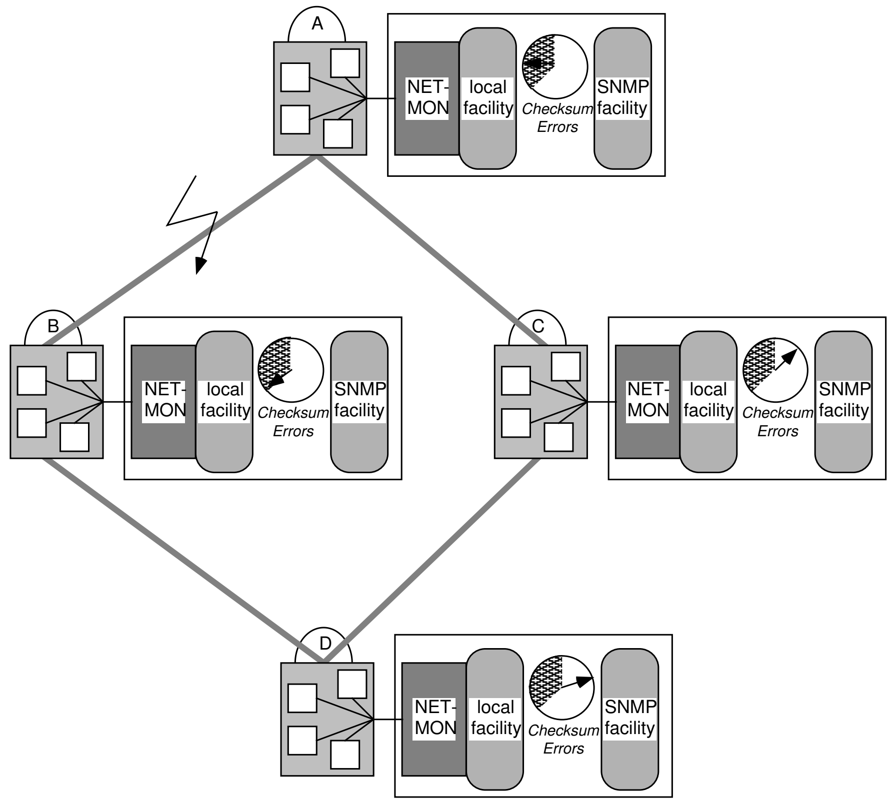

Figure 5: Example for a scenario with a SNMP facility acting as a Manager

6 Conclusion

Within this paper we described how parallel communication architectures can be monitored and managed efficiently by supplying an autonomous working device for this purpose. Unlike in [RSZ90] this approach uses SNMP-2 as management protocol due to its simplicity and broad availability. The elder approach to use OSI management has been abandoned because of two reasons: First, the implementation of an OSI-Agent requires a running OSI-stack which puts additional load (concerning CPU and memory) on the management entity. Second, there are (almost) no management applications available to cope with OSI-Agents. Another item we have added to the management station is the possibility of acting as a Manager as well as an Agent. Thus, not only hierarchical but also really distributed management is possible, since local management activities can be fulfilled by local management entities and no central entity is involved.

The practicability of the described approach will be shown by realizing the management entity to be used to monitor and manage a transputer network. Therefore, three students are currently working on the development the NETMON interface card [Skr93], the NETMON monitoring software [Mai93a], the NETMON control station's software [Mai93b] and the local as well as the SNMP facility [Rot93]. The results of these works are expected to be published as diploma thesises in the fall of 1993.

The ongoing work will try to develop rules for the local facility to gain decentralization and to improve the effectiveness of the management entity.

References

[Bau93] H.G. Baumgärtel. OSI-Standard ASN.1. *IX – Multiuser Multitasking Magazin*, 1993(3):118 – 126, March 1993.

[BZ92] T. Braun and M. Zitterbart. Parallel transport system design. In 4^{th} *IFIP Conference on High Performance Networking (HPN92)*, Lüttich, Belgium, December 14 – 18 1992.

[CFSD90] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A simple network management protocol (SNMP). Network Working Group, Request for Comments 1157, May 1990.

[Gla92] M. Glander. NETINT, version 1.5. Internal report, Institute of Telematics, University of Karlsruhe, October 1992.

[Jan92] M. Jander. Coming soon to a network near you. *Data Communications*, 21(16):66 – 76, November 1992.

[Mai93a] T. Maier. Entwurf und Implementierung der Monitorsoftware für NETMON-III. Studienarbeit, University of Karlsruhe, Institute of Telematics, June 1993.

[Mai93b] T. Maier. Konzipierung und Realisierung der NETMON-III Leitstation. Diplomarbeit, University of Karlsruhe, Institute of Telematics, December 1993.

[MR88a] K. McCloghrie and M.T. Rose. Management information base for network management of TCP/IP-based internets. Network Working Group, Request for Comments 1066, August 1988.

[MR88b] K. McCloghrie and M.T. Rose. Structure and identification of management information for TCP/IP-based internets. Network Working Group, Request for Comments 1065, August 1988.

[MR91] K. McCloghrie and M.T. Rose. Management information base for network management of TCP/IP-based internets: MIB-II. Network Working Group, Request for Comments 1213, March 1991.

[RM90] M.T. Rose and K. McCloghrie. Structure and identification of management information for TCP/IP-based internets. Network Working Group, Request for Comments 1155, May 1990.

[RM91] M.T. Rose and K. McCloghrie. Concise MIB definitions. Network Working Group, Request for Comments 1212, March 1991.

[Rot93] L. Roth. Auf SNMP basierendes Managementsystem für eine parallele Kommunikationsarchitektur. Diplomarbeit, University of Karlsruhe, Institute of Telematics, September 1993.

[RS92] O. Rose and J. Seitz. Management von Kommunikationsnetzen. Tutorium, GI-Jahrestagung 1992, Karlsruhe, Federal Republic Of Germany, September 28 1992.

[RSZ90] O. Rose, J. Seitz, and M. Zitterbart. An architecture for efficient OSI network management. In *LAN Proceedings – The Seventh Annual European Fibre Optic Communication and Local Area Networks Exposition, Park Hilton, Munich, Federal Republic Of Germany*, pages 133 – 138, 214 Harvard Avenue, Boston, MA, 02134 U.S.A., June 27 – 29 1990. IGI Europe, a Division of Information Gatekeepers, Inc.

[Skr93] S. Skrodzki. Entwurf und Realisierung der NETMON-III Interface Karte. Diplomarbeit, University of Karlsruhe, Institute of Telematics, October 1993.

[Zit91] M. Zitterbart. *Funktionsbezogene Parallelität in transportorientierten Kommunikationsprotokollen.* Number 183 in 10: Informatik/Kommunikationstechnik. VDI Verlag, 1991.