Es erlaubt umfangreiche Prozessanwendungen auf verteilten Systemen in einfacher und effizienter Weise zu implemetieren und wird derzeit bei der Realisierung verschiedener Projekte eingesetzt.

## 2. Das KAE-Botschaftensystem:

Das KAE-Botschaftensystem stellt einen wirkungsvollen Satz von Funktionen zur Koordinierung und Synchronisation paralleler Rechenprozesse in Verbindung mit Pufferung und Zeitueberwachung von Botschaften, welche zwischen Sende- und Empfangsprozessen uebertragen werden, bereit.
Das System erlaubt die Kommunikation von Rechenprozessen:

- eines Benutzers
- verschiedener Benutzer eines Rechners
- auf Betriessystemebene
- zwischen Benutzern und Betriebssystem
- von Rechner zu Rechner unter Nutzung vorhandener niederer Netzwerkebenen

Um fuer kritische Kommuniaktionwege, welche zwischen Rechenprozessen auf Systemebene gedacht sind, ein Einbrechen von Benutzern in den Botschaftenverkehr zu verhindern, koennen diese Kommunikationswege fuer nicht privilegierte Benutzer gesperrt werden.

### 2.1 Gliederung nach Funktionsbereichen

Das Botschaftensystem besteht aus den Teilsystemen:

- QUEUEING-System ( B i l d 1 )

und

- SCHEDULING-SYSTEM ( B i l d 2 )

sowie einer

- DIALOGSCHNITTSTELLE

Das Queueing-System dient zur Organisation des Botschaftenvekehrs auf der der Basis zyklischer bzw. nicht zyklischer Queues nach dem First-in/First-out-Prinzip.
( B i l d 3 )
Durch Eroeffnen einer zentralen zyklischen Beobachtungsqueue besteht die Moeglichkeit, fuer angewaehlte Queues eine zentrale Verfolgung der Einschreibvorgaenge zu taetigen.
Daneben ist optional ein Roll-in/Roll-out-Konzept mit optimierter, stossfreier Nachschubsteuerung zur Nutzung von Externspeichern vefuegbar. ( B i l d 4 )

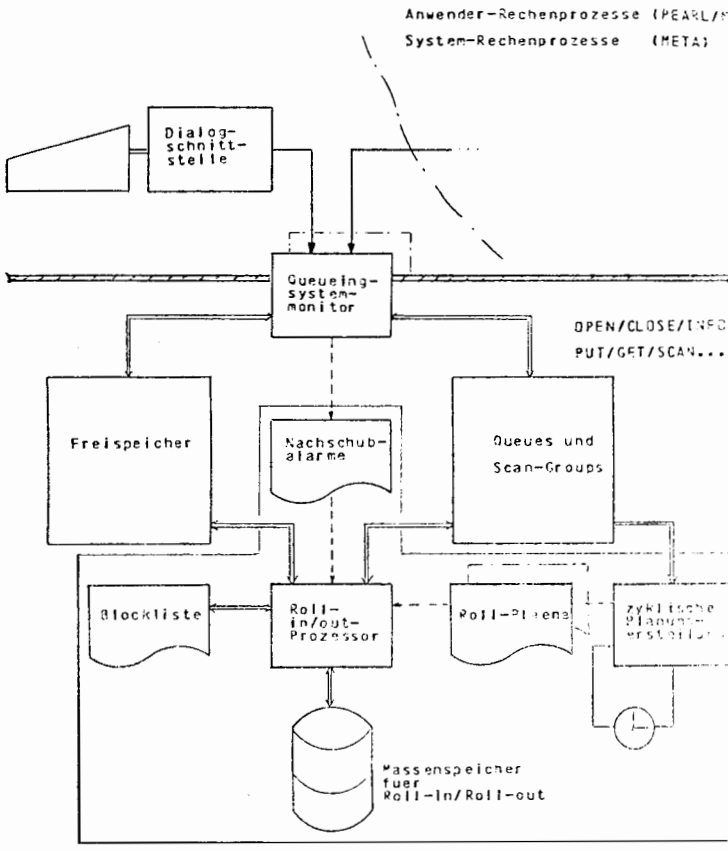


Bild 1. QUEUEING-SYSTEM mit optionaler Massenspeicherorganisation

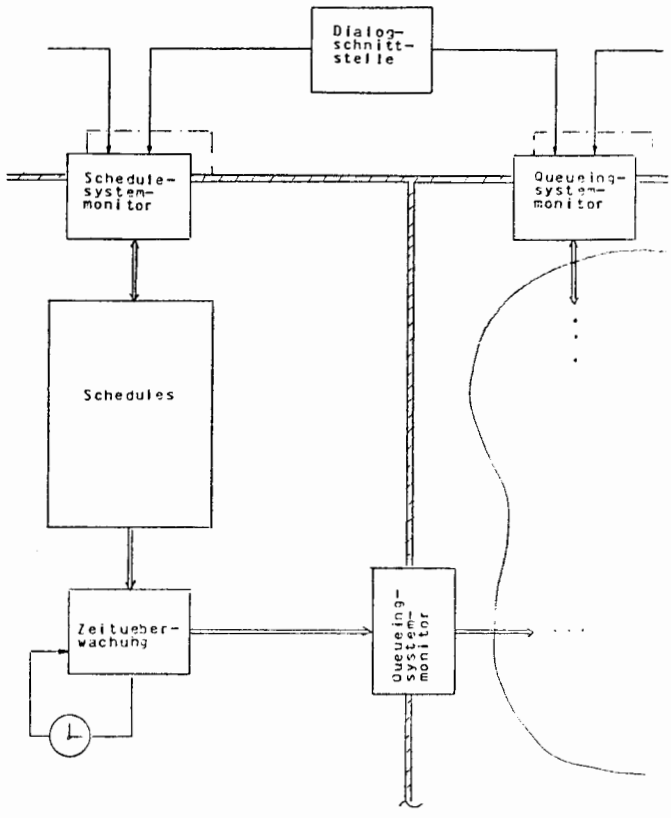


Bild 2. SCHEDULING-SYSTEM mit Zugang zum QUEUEING-System

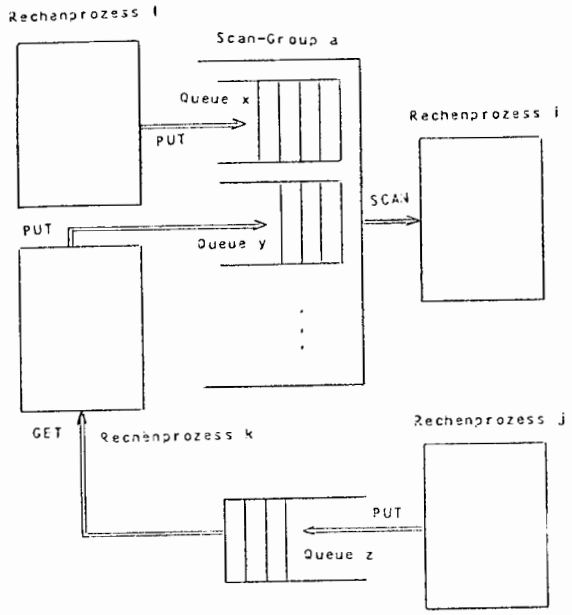


Bild 3. Rechenprozesskommunikation ueber QUEUEs und SCAN-GROUPs

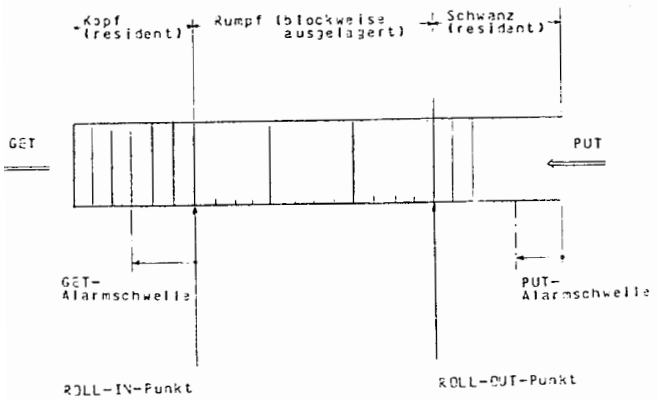


Bild 4. Abschnitte einer QUEUE mit Massenspeicherverdraengung

Das Scheduling-System dient zur zeitlichen Ueberwachung komplexer Abhaengigkeiten zwischen Anstoessen und Reaktionen in einem Prozess auf der Basis zeitueberwachter Botschaften, die unter Angabe von Schluesseln aufgegen und abgerufen werden. Im Falle von Ueberschreitung vorgegebener Verweildauern dieser Botschaften in einer Schedule-Liste koennen entsprechende Botschaften in dafuer optional angebbare Fifo-Queues einsgespeist werden.

Die Dialogschnittstelle dient zur manuellen bzw. per Kommando-Files gesteuerten Durchfuehrung von Einstell- und Protokollierungsfunktionen. Alle Funktionen, welche in der Kommandosprache zur Verfuegung stehen, sind auch ueber Programmschnittstellen nutzbar. Insbesondere dient die Dialogschnittstelle zum Ein/Ausschalten von Trace-optionen sowie der Erzeugung von Uebersichtsprotokollen ueber Belungszustaende und Auslastungen im lfd. Betrieb.

## 2.2 Einbindung in Sprachen

Die Benutzung des Botschaften-Systems wird fuer die Realzeitsprachen:

- META und PEARL

unterstuetzt. Zur effektiven Anbindung der Funktionen an PEARL wurde folgendes unternommen:

- Einfuehrung von Call-Prozeduren, welche dem Compiler per spezieller globaler ENTRY-Deklarationen bekannt gemacht werden und so die Code-Generierung auf die fuer das Botschaftensystem notwendigen Schnittstellen steuern.

- Einfuehrung von Signalanschluessen fuer Fehlerreaktionen aus dem Botschaftensytem.

- Einfuehrung von Equivalenzen, um alternative Sichten auf Botschaften zu gestatten. Die Anwendung dieser Methode wird nicht auf Bereiche beschraenkt, welche als Ports fuer Botschaften dienen, sondern kann auch fuer interne Datenmanipulationen benutzt werden. Die Equivalenzanweisung ueberlagert eindimensionale Bereiche vom vom Typ FIXED linksbuendig mit beleibigen nulldimesionalen Strukturen bzw. einfachen PEARL-Objekten (FIXED, FLOAT, CHAR, DUR, ...).

Die obigen Einschraenkungen gegeueber der ausufernden Benutzung von EQUIVALENCE in FORTRAN dient dazu, die Pruefbarkeit der Bereichsgrenzen in Normalfall zur Compile-Zeit zu gewaehrleisten und fuer den Fall von per Adresse uebergebenen Bereiche eine Pruefung zu Laufzeit zu sichern.
In keinem Falle koenen durch das gewaehlte Verfahren angrenzende Daten angegriffen werden.
Um ein Gefuehl fuer die Nutzung der des Botschaftensystems in PEARL zu vermitteln, wird hier das Beispiel eines Empfangsprozesses gegeben, welcher aus einener Queue Alarmmeldungen bzw. Messwerte ausliest und weiterleitet:

```
RECEIVE: TASK;
 DCL QNR FIXED INIT(10);
 DCL LNG FIXED INIT(20);
 DCL STV STRUCT [(STATUS,EFFLNG,NRQ) FIXED];
 DCL BUFFER(20) FIXED;
 EQU BUFFER: TYP FIXED,
     MELDUNG STRUCT [TYP FIXED, WERT BIT(2)],
     MESSUNG STRUCT [TYP FIXED, WERT FLOAT];

ON QSERROR: BEGIN CALL ERROR(QNR);
            RETURN; END;

REPEAT
CALL GET_REQUEST (QNR, STV, BUFFER, LNG);
IF TYP EQ 1
THEN CALL ALARM (MELDUNG.WERT);
ELSE CALL CHECK(MESSUNG.WERT);
FIN;
END;

END;
```

## 2.3 Basisstrukturen des QUEUEING-Systems

Die Funktionen des KAE-Botschaftensystems beziehen sich auf 3 Arten von Datenstrukturen: Queues, Scan-Groups und Schedules.

### 2.3.1 Queues

Dieses sind nach First-in/First-out-Prinzip organisierte Puffer fuer Botschaften, welche eine im Rahmen des fuer das Queueing-System konfigurierten Haupt- und Massenspeichers beliebige Anzahl von Botschaften zwischenspeichern koennen.
Dabei ist es moeglich, das in einer Queue unterschiedliche Botschaftstypen vorgebbarer Maximallaenge eingespeist werden.
Besonders hervorzuheben ist die Moeglichkeit der Eroeffnung einer Eroeffnung einer BEOBACHTUNGS-QUEUE.
Sie ermoeglicht es, optional Botschaften, welche in Queues eingespeist werden, unter Angabe von Uhrzeit, Ziel-Queue, Status und Absenderidentifikation mitzuschreiben.
Somit wird es moeglich, durch Auslesen der Beobachtungs-Queue selektive Beobachtungen von Botschaftenstroemen zentral vorzunehmen.
Um die Rueckwirkungen dieser Protokollierung auf die Dynamik des Botschaftenverkehrs minimal halten, ist die Beobachtungs-Queue als zyklische Queue ausreichender Kapazitaet eroeffnen.
Wird die Pufferkapazitaet dieser zyklischen Queue ueberschritten, so werden die jeweils aeltesten Botschaften durch neu anfallende ueberschrieben.
Es sei an dieser Stelle auf die Moeglichkeit hingewiesen, durch einfaches Umspeichern von der Beobachtungs-Queue in eine Queue mit der Moeglichkeit der Externspeicherverdraengung, auch groessere Historien aufzunehmen.

### 2.3.2 Scan-Groups

Sie sind der Zusammenschluss mehrerer Queues zu einer Eingabegruppe, welche an den aufrufenden Empfangsprozess die erste Botschaft uebermittelt, die beim Abtasten der Queues entsprechend deren Rangfolge gefunden wird.
Es gibt Scan-Groups mit fester Rangfolge und solche, bei denen alle Queues gleichberechtigt sind.
Queues, die z. Zt. Mitglied einer Scan-Group sind, nicht gleichzeitig anderen Scan-Groups zugeteilt werden.

### 2.3.3 Schedules

Unter Schedules sind hier solche Botschaften zu verstehen, welche einer Zeituebewachung unterliegen und hauptspeicherresident sind.
Das Aufgeben (SCHEDULING) dieser Botschaften erfolgt mit Bezug auf eine Schluesselklasse, aus welcher hervorgeht, welcher Teil der Botschaft als Schluesselbegriff fuer das Aufsuchen und Befreien (DISPATCHING) vor Ablauf der Ueberwachungszeit dient.
Botschaften, die wegen der Ueberschreitung der Zeitvorgaben durch die Zeitueberwachung befreit werden, koennen optional an dafuer vorgesehene Queues weitergeleitet werden.

2.4.1 PEARL-Aufrufe zum QUEUEING-System

Die Aufrufe an das QUEUEING-Systems gliedern grob in die Funktionsbereiche:

- Einstellungen (OPEN/CLOSE/TRACING)
- Auskuenfte (INFO)
- Botschaftentransfer (PUT/GET/SCAN)

Bei Ausfuehrung der Funktionen PUT, GET und SCAN des Queueing-Systems koennen beim Einschreiben bzw. Auslesen Nachschubprobleme bzgl. der Verfuegbarkeit von Freispeicher (PUT) bzw. des Vorhandensein von Botschaften in Queues (GET/SCAN) temporaer auftreten. Dabei werden Nachschubprobleme, welche auf das Konto Externspeicherverdraengung gehen, durch eine praeventives ROLL-IN/ROLL-OUT-Strategie minimiert.
Fuer kritische Queues koennen bei Eroeffnung sowohl Speichergarantien als auch Roll-out-Sperren definiert werden.
Es ist nun fuer den Anwender sinnvoll, im Falle von Nachschubproblemen, jeweils selbst selbst zu entscheiden, welche der drei folgenden Verhaltensweisen in Verbindung mit dem lfd. PUT, GET oder SCAN zweckmaessig ist:

- CALL ..._REQUEST (...);
Diese Methode versetzt den aufrufenden Rechenprozess in den Zustand "blockiert", bis das Nachschubproblem geloest ist.

- CALL ..._TIMEOUT (...);
Hier wird der aufrufende Rechenprozess in den Zustand "blockiert" versetzt, bis entweder das Nachschubproblem geloest oder die im CALL angegebene Zeitdauer ueberschritten ist. Der befreite Rechenprozess hat dann die Moeglichkeit durch ON-Reaktion bzw. Statusabfrage Erfolg oder Misserfolg festzustellen.

- CALL ..._TRY (...);
Mit dieser Methode versucht der aufrufende Rechenprozess die Eingabe bzw. Ausgabe. Er bekommt ohne implizite Synchronisation (ggf. warten auf Nachschub) die Kontrolle zurueck und kann sich ebenfalls per ON-Reaktion oder Statusabfrage ueber das Ergebnis informieren.

Die PEARL-Schnittstellen sind im folgenden mit kurzer Erlaeuterung der verwendeten Parameter aufgefuehrt.

Nummer der Queue:
QNR FIXED

Nummer der Scan-Group:
GNR FIXED

Maximale Transferlaenge. Die Ueberpruefung auf Vertraeglichkeit mit der Transferpuffergroesse geschieht zur Compile-Zeit und ggf. zur Laufzeit:
LNG FIXED

Zur Erhoehung der Transparenz des Botschaftenverkehrs mittels Beobachtungs-Queue, kann der Absender einer Nachricht optional eine Aosenderidentifikation angeben:
IDF FIXED

Statusrueckinformation mit den Komponenten Status bezogen auf den letzten Aufruf, effektive Transferlaenge und (fuer Lesen aus einer Scan-Group) Queuenummer, aus der die erhaltene Botschaft stammt:
STV STRUCT [(STATUS,EFFLNG,EFFQNR) FIXED]

Sende/Empfangspuffer fuer Botschaften:
MES STRUCT[beliebig ] bzw.
MES(n) FIXED

Empfangspuffer fuer eingeholte Informationen:
INF STRUCT[n mal FIXED ] bzw.
INF(n) FIXED

Puffer zur Aufnahme von Queuenummern einer Scan-Group:
QVC STRUCT[n mal FIXED ] bzw.
QVC(n) FIXED

Parameterpuffer fuer die Eroeffnung von Queues:
```
PAR STRUCT[(QUEUETYP,MAXMESSAGELNG,CAPACITY,
        PUTALARMLIMIT,GETALARMLIMIT) FIXED]
```

Relative Zeitvorgabe:

DUR DURATION

Eroeffnen einer Queue:
CALL OPEN_QUEUE(QNR,STV,PAR);

Schliessen einer Queue falls diese leer ist:
CALL CLOSE_QUEUE(QNR,STV);

Loeschen aller Botschaften einer Queue:
CALL CLEAR_QUEUE(QNR,STV);

Loeschen aller Botschaften einer Queue und Schliessen derselben:
CALL KILL_QUEUE(QNR,STV);

Mitschreiben in Beobachtungs-Queue anwaehlen:
```
CALL TRACE_ON(QNR,STV);
```

Mitschreiben in Beobachtungs-Queue abwaehlen:
```
CALL TRACE_OFF(QNR,STV);
```

Statische und dynamische Information ueber eine Queue anfordern:
```
CALL INFO_QUEUE(QNR,STV,INF,LNG);
```

Auslesen von Botschaften aus einer Queue:
```
CALL GET_REQUEST(QNR,STV,MES,LNG );
CALL GET_TIMEOUT(QNR,STV,MES,LNG,DUR);
CALL GET_TRY(QNR,STV,MES,LNG);
```

Einschreiben von Botschaften in eine Queue:
```
CALL PUT_REQUEST(QNR,STV,MES,LNG[,IDF]);
CALL PUT_TIMEOUT(QNR,STV,MES,LNG[,IDF],DUR);
CALL PUT_TRY(QNR,STV,MES,LNG[,IDF]);
```

Eroeffnung einer Scan-Group aus Mitglieds-Queues, welche gleichrangig sind:
```
CALL CYCLIC_GROUP(GNR,STV,QVC,NMQ);
```

Eroeffnen einer Scan-Group aus Mitglieds- mit Vorrangsregelung:
```
CALL PRIORITY_GROUP(GNR,STV,QVC,NMQ);
```

Aufloesen einer Scan-Group:
```
CALL CLOSE_GROUP(GNR,STV);
```

Abfragen des Typs sowie der Zusammensetzung Scan-Group:
```
CALL INFO_GROUP(GNR,STV,INF,LNG);
```

Botschaften aus einer Scan-Group lesen:
```
CALL SCAN_REQUEST(GNR,STV,MES,LNG);
CALL SCAN_TIMEOUT(GNR,STV,MES,LNG,DUR);
CALL SCAN_TRY(GNR,STV,MES,LNG);
```

Funktionen bezogen auf die Beobachtungs-Queue in Analogie zu den entsprechenden Funktionen auf normale Queues:
```
CALL OPEN_TRACE(STV,PAR);
CALL CLOSE_TRACE(STV);
CALL CLEAR_TRACE(STV);
CALL KILL_TRACE(STV);
CALL INFO_TRACE(STV,INF,LNG);
CALL GET_TRACE_REUQEST(STV,MES,LNG);
CALL GET_TRACE_TIMEOUT(STV,MES,LNG,DUR);
CALL GET_TRACE_TRY(STV,MES,LNG);
```

Einholen von statischen und dynamischen Parametern bzgl. des Gesamtsystems wie z. b. Anzahl der Queues, Speicherbelastung, Massenspeicherbelastung, u. s. w. :
```
CALL GENERAL_INFO_QS (STV,INF,LNG);
```

### 2.4.2 PEARL-Aufrufe zum SCHEDULING-System

Die Aufrufe lassen sich in die Bereiche:
- Einstellungen (OPEN/CLOSE)
- Auskuenfte (INFO)
- Ein/Ausplanungen von Botschaften (SCHEDULE/DISPATCH)

gliedern.

Die Schnittstellen zu PEARL sind nachfolgend mit kurzer Erlaeuterung der verwendeten Parameter aufgefuehrt.

Nummer der angewaehlten Schluesselklasse:
```
KNR FIXED
```

Nummer der Queue, in welche die Botschaft im Falle von Zeitueberschreitung eingespeist wird. (negativer Wert bewirkt Wegwerfen):
```
QNR FIXED
```

Transferlaengenvorgabe. Sie begrenzt die Laenge der uebertragenen Information nach oben und wird zur Compile-Zeit oder ggf. zur Laufzeit auf Vertraeglichkeit mit der Transferpuffergroesse abgeprueft:
```
LNG FIXED
```

Statusrueckinformation mit den Komponenten Status und der effektiven Transferlaenge:
```
STV STRUCT[(STATUS,EFFLNF) FIXED]
```

Schluesseldeskriptor fuer eine Schluesselklasse, der die Bitslices angibt, welche in einer Botschaft als Schluesselbegriffe verwendet werden:
```
KEY STRUCT[n mal FIXED ] bzw.
KEY(n) FIXED
```

Sende/Empfangspuffer fuer ein zeitueberwachte Botschaften:
```
SCD STRUCT[beliebig  ] bzw.
SCD(n) FIXED
```

Empfangspuffer fuer eingeholte Auskuenfte:
```
INF STRUCT[n mal FIXED ] bzw.
INF(n) FIXED
```

Zeitvogaben:

| | | |
|---|---|---|
| TIME | | absolute Zeit |
| DELAY | DURATION | relative Zeit |
| DAYTM | CLOCK | Tageszeit |

Eroeffnen einer Schluesselklasse:
```
CALL OPEN_KEY(KNR,STV,KEY,LNG);
```

Schliessen einer Schluesselklasse:
```
CALL CLOSE_KEY(KNR,STV);
```

Loeschen aller Schedules einer Schluesselklasse:
CALL CLEAR_KEY(KNR,STV);

Schliessen einer Schluesselklasse nach vorherigem Loeschen aller enthaltenen Schedules:
CALL KILL_KEY(KNR,STV);

Statische und dynamische Information ueber eine Schluesselklasse einholen:
CALL INFO_KEY(KNR,STV,INF,LNG);

Suchen und ggf. befreien eines Schedules:
CALL FIND_DISPATCH (KNR, STV, SCD, LNG, TIME);

Einplanen eines Schedules mit Zeitvogaben:
CALL ...
SCHEDULE_DURING(KNR,STV,SCD,LNG,DELAY,QNR);
SCHEDULE_UNTIL(KNR,STV,SCD,LNG,DAYTM,QNR);
SCHEDULE_TIME(KNR,STV,SCD,LNG,TIME,QNR);
SCHEDULE_FOREVER(KNR,STV,SCD,LNG,QNR);

Einholen statischer und dynamischer Informationen ueber das Gesamtsystem:
CALL GENERAL_INFO_SS(STV,INF,LNG);

## 3. Benutzung des Queueing-Systems fuer die rechneruebergreifende Kommunikation von Rechenprozessen

Mit Hilfe der geschilderten Grundfunktionen lassen sich Systemprozessoren (Software, Firmware,Hardware) zur Realisierung rechneruebergreifender Kommunikation auf der Basis niederer Netzwerk-Protokolle in einfacher Weise benutzen. Das bedeutet der Anwender schreibt eine Botschaft in eine der Queues, ueber welche ein solcher Link-Prozessor beauftragt wird und gibt in diser Botschaft an definierter Stelle die Nummer der realen bzw. virtuellen Verbindung sowie die Queue des Empfangsrechners als Adresse an. Optional wird ein Absenderfeld als Teil der Botschaft kodiert. Die Realisierung eines Protokolls entsprechend den jeweiligen Projektanforderungen ist dabei expliziter Bestandteil der Anwenderrechenprozesse und kann in einer Sprache wie PEARL klar und uebersichtlich formuliert werden.
Das Problem, welches es jedoch zu loesen gibt, ist eine rechnernetzgerechte zuordnung von Rechenprozessen zu Queues. Mit anderen Worten: Es muss eine Institution geschaffen werden, ueber welche bezogen auf das Gesamtnetz folgende Auskuenfte eingeholt werden koennen:

- Ueber welche Queues (lokal) erreiche man welchen Link-Prozessor ?
  Der jeweilige Link-Prozessor kann dann seinerseits per Botschaft Auskunft ueber die von ihm verwalteten Links geben.
- Welche Rechenprozesse und Dienstleistun- koennen ueber lokale Queues bzw. solche, welche ueber bestimmte Links erreichbar sind angesprochen werden ?

Zur Realisierung dieser Funktion ist in jedem Rechner ein QUEUE-INFO-Prozessor zu vorzusehen, welcher ueber eine feste Queue 0 Auftraege entgegennimmt, die folgende Funktionen zum Inhalt haben:

- Eintraege ins "SCWARZE BRETT", welche die Empfangsprozesse bzw. Dienstleistungen betreffen, die ueber eine gegebene Queue Scan-Group erreichbar sind.
- Auskuenfte vom "SCHWARZEN BRETT" im Sinne von queuebezogenen bzw. funktionsbezogenen Auskuenften. ( B i l d 5 )

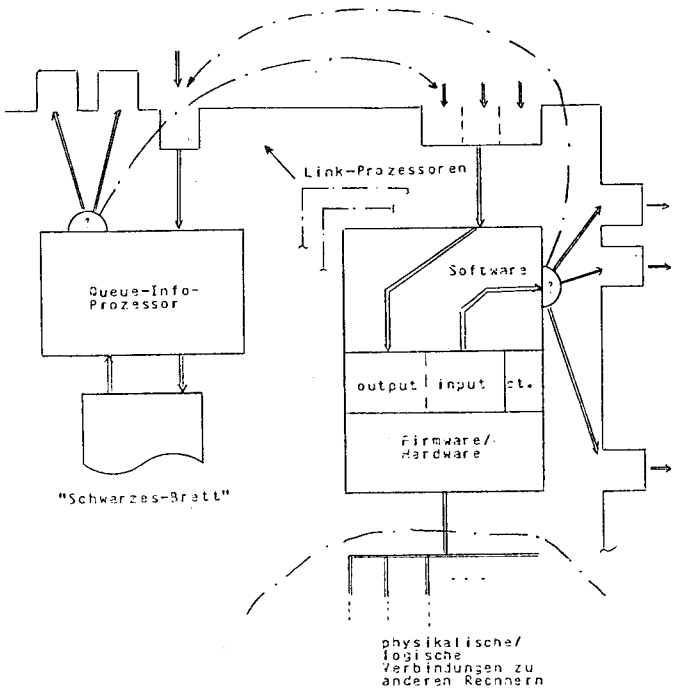


Bild 5. Link-Prozessoranschluesse mit Auskunftssystem

## 4. Schlussbemerkungen

Der seitens KAE gewaehlte Ansatz zur Nutzung von Botschaftskonzepten fuer verteilte Systeme geht von Botschafts- und Ueberwachungsmechanismen aus, welche so oder so aehnlich in modernen Prozessrecnersystemen vorhanden sind. Das bedeutet, dass die Kommunikationsstrukturen in PEARL zwar genutzt, jedoch nicht definiert werden. Es macht auch durchaus einen Sinn, die Lebensdauer von Objekten wie z. b. Queues nicht an die Lebensdauer des jeweiligen Empfangsprozesses zu binden da bei groesseren Anwendungen mit 24-Stundenbetrieb einen "fliegenden Austausch" von Software im lfd. Betrieb erforderlich sein kann.

Ein weiterer Gesichtspunkt fuer den Ansatz ist, dass gerade in verteilten Systemen, die Kommunikationsstruktur nicht der Semantik nur einer Sprache gerecht werden darf, sondern ein Konzept zu waehlen ist, welches sich in einfacher Weise auch in andere Sprachen einbringen laesst.

Bockhoff, Werner
KAE Bremen, Sebaldsbruecker Heerstr. 235
tel. 0421/ 4572398