Aufgabe 2: A Link Protocol (ALP)

[1] Fletcher, J. G.
Serial Link Protocol Design
Computer Communication Review, Vol. 14, No. 2, 1984, pp. 26-33

a) Wie einigen sich zwei Knoten nach der Initialisierung auf einen gemeinsamen Zustand?

b) Wie erfolgt der Datentransfer?

c) Was sind die Grundideen dieses Algorithmus?

Lösungsvorschlag Aufgabe 2: A Link Protocol (ALP)

a) Wie einigen sich zwei Knoten nach der Initialisierung auf einen gemeinsamen Zustand?

- Für die Sequenznummern können z.B. 4 bit vorgesehen werden, damit gilt m=16.
- Der Initialzustand sieht laut Protokollspezifikation wie folgt aus ("*" bedeutet dabei eine beliebige Initialisierung):

Knoten A
(nach der Initialisierung)

Name	Wert
oqueue	leer
ophase	0*
oseqno	1*
ocount	0
owindow	0
olength	0
iphase	1*
iseqno	0*
iwindow	5
ireply	1

Knoten B
(nach der Initialisierung)

Name	Wert
oqueue	leer
ophase	1*
oseqno	1*
ocount	0
owindow	0
olength	0
iphase	1*
iseqno	1*
iwindow	3
ireply	1

- Angenommen Knoten A sendet zuerst. Er erzeugt dazu ein leeres Paket, in dem alle Werte auf 0 vorbesetzt sind und das keine Daten enthält, danach werden einige Felder aufgrund des Algorithmus gefüllt, Knoten B empfängt das Paket und es ergibt sich folgende Situation:

Knoten A

Name	Wert
oqueue	leer
ophase	0*
oseqno	1*
ocount	0
owindow	0
olength	0
iphase	1*
iseqno	0*
iwindow	5
ireply	1->0

Name	Wert
dreply	0
dphase	0*
dseqno	1*
rwindow	5
rphase	1*
rseqno	0*
dpacket	0
data	leer

Knoten B

Name	Wert
oqueue	leer
ophase	1*
oseqno	1*
ocount	0
owindow	0
olength	0
iphase	1->1
iseqno	1*
iwindow	3
ireply	1->1

- Das einzige was in diesem Fall geändert wird, ist die i-Phase des Knoten B auf die konträre o-Phase des Knotens A. Damit wird eine erneute Synchronisation erzwungen.

- Knoten B muß im Gegenzug ein Kontrollpaket senden.

- Da bei Knoten B *ireply* gesetzt ist, schickt er eine Antwort zurück.

Knoten A

Name	Wert
oqueue	leer
ophase	0->1
oseqno	1->1
ocount	0->0
owindow	0->3
olength	0
iphase	1->0
iseqno	0*
iwindow	5
ireply	0->1

Name	Wert
dreply	0
dphase	1*
dseqno	1*
rwindow	3
rphase	1
rseqno	1*
dpacket	0
data	leer

Knoten B

Name	Wert
oqueue	leer
ophase	1*
oseqno	1*
ocount	0
owindow	0
olength	0
iphase	1
iseqno	1*
iwindow	3
ireply	1->0

- Weiterhin muß eine Antwort von A nach B geschickt werden, da die Phasen (jetzt in der Gegenrichtung) nicht übereinstimmen.

Knoten A

Name	Wert
oqueue	leer
ophase	1
oseqno	1
ocount	0
owindow	3
olength	0
iphase	0
iseqno	0*
iwindow	5
ireply	1->0

Name	Wert
dreply	0
dphase	1
dseqno	1
rwindow	5
rphase	0
rseqno	0*
dpacket	0
data	leer

Knoten B

Name	Wert
oqueue	leer
ophase	1->0
oseqno	1->0
ocount	0->0
owindow	0
olength	0
iphase	1
iseqno	1*
iwindow	3
ireply	0

b) Wie erfolgt der Datentransfer?

- Angenommen, B erhält eine Nachricht und soll sie an A senden. Dann wird laut Regel (1) die Nachricht in die Warteschlange *oqueue* eingetragen und *olength* erhöht. Dann wird nach Regel (7) ein Paket an A gesendet, das aber noch keine Daten enthält, da *owindow* immer noch 0 ist, aber bei dem *dreply* auf 1 gesetzt ist. Das veranlaßt A ein weiteres Paket zu senden, mit Hilfe dessen Knoten B endlich *owindow* auf 5 setzen kann. Knoten B kann nun das unten dargestellte Datenpaket senden und Knoten A empfängt es. In diesem Fall sind also inklusive Initialisierungsphase 6 Nachrichten ausgetauscht worden.

Knoten A

Name	Wert
oqueue	leer
ophase	1
oseqno	1
ocount	0
owindow	3
olength	0
iphase	0
iseqno	0->1
iwindow	5
ireply	0->1

Datenpaket ←

Name	Wert
dreply	1
dphase	0
dseqno	0
rwindow	3
rphase	1
rseqno	1
dpacket	1
data	**N1**

Knoten B

Name	Wert
oqueue	N1
ophase	0
oseqno	0
ocount	0->1
owindow	0->5
olength	1
iphase	1
iseqno	1
iwindow	3
ireply	0

- Knoten A muß noch eine Quittung auf das Paket schicken. Das muß aber nicht sofort sein, sondern er kann noch abhängig von der Fenstergröße weitere Pakete empfangen und dann mehrere gemeinsam bestätigen.

c) Was sind die Grundideen dieses Algorithmus?

- Es handelt sich um eine Zustandsaustauschmethode.
- Die Nachrichten werden mit Sequenznummern durchnummeriert.
- Quittungen werden nur von Zeit zu Zeit gesendet (kein Alternating Bit Protokoll).
- Die Quittungen können an Nachrichten in die Gegenrichtung gekoppelt sein.
- Eine Störung leitet eine neue Epoche ein. Es genügt dabei, Epochen mit {0,1} zu kodieren.


```
// This algorithm should be executed after
// receipt of each non-erroneous frame.
// ein neues Paket trifft ein
{  f = receive();

    // Das empfangene Paket ist das erwartete,
    // d.h. Die In-Phase und die
    // erwartete Sequenznummer stimmen überein.
  if (f->dseqno == iseqno
      && f->dphase == iphase) {

        // Es handelt sich um ein Datenpaket und
        // das Eingangsfenster ist geöffnet.
    if (f->dpacket == 1 && iwindow > 0) {

      // An input packet has arrived
      // in sequence.Accept it

        // Das eingehende Paket wurde akzeptiert,
        // die Sequenznummer für das nächste Paket
        // wird eingestellt.
      iseqno = (iseqno + 1) % m;

        // Die daten werden an die nächsthöhere
        // Schicht weitergeleitet
      forward(f->data);
    }
    // Bei einer reinen Kontrollnachricht
    // geschieht nichts.

  } else {
    // An input packet has been lost.
    // Prepare to accept retransmission.
    // Es ist ein Fehler aufgetreten, eine neue
    // Epoche muß eingeleitet werden.

    iphase = 1 - f->dphase;
    ireply = 1;
  }

    // Das Paket enthält eine Bestätigung für von
    // mir gesendete Pakete und bei mir gibt es
    // noch unbestätigte Pakete, die gesendet
    // wurden
  if ((f->rseqno - oseqno) % m <= ocount) {
    // The received reverse sequence number
    // is not anomalous.

    while (f->rseqno <> oseqno) {
```

```
      // Discard accepted output packets.

        // Entfernen des Pakets aus meiner Ausgangswarteschlange
        pop(oqueue);

        // Sequenznummer für bestätigte Pakete erhöhen
        oseqno = (oseqno + 1) %m;

        // Zähler für gesendete , aber noch nicht bestätigte
        // (=ausstehende) Pakete erniedrigen
        ocount--;

        // Zähler für zu übertragende Pakete erniedrigen
        // d.h. Länge der Output-Queue erniedrigen
        olength--;
    }

    // Die eventuell geänderte Fenstergröße des
    // Kommunikationspartners übernehmen
    owindow = f->rwindow;
  }

    // Die Phase in meiner Senderichtung stimmt nicht mehr
    // überein mit der Sicht des Kommunikationspartners.
  if (f->rphase <> ophase) {

    // Prepare to retransmit rejected output packets

    // Phase und erwartete Sequenznummer des
    // Kommunikationspartners übernehmen
    ophase = f->rphase;
    oseqno = f->rseqno;

    // Alle bisher übertragenen und noch nicht quittierten
    // Nachrichten sind ungültig und
    // müssen neu übertragen werden.
    ocount = 0;
  }

    // Der Kommunikationspartners erwartet eine Antwort oder
    // ich muß Daten senden.
  if (f->dreply == 1 || olength > 0) {
    // state is unsatisfactory.
    ireply = 1;
  }
    // Wegwerfen des Frames
  discard(f);
}
```



```
// This algorithm should be executed
// (1) after a packet is pushed on oqueue and olength
//     is incremeted and
// (2) after execution of the preceding algorithm for
//     receipt of a non-erronous frame.

// Also ireply should be set to 1 and this algorithm
// then executed
// (3) after receipt of an erroneous frame,
// (4) after iwindow is changed,
// (5) after a give-up timeout and any associated
//     purging of the output queue,
// (6) after initialization, and perhaps
// (7) periodically so long as olength > 0.


// Es könen noch Nachrichten innerhalb des Fensters gesendet
// werden und es sind gleichzeitig noch ungesendete in der
// Warteschlange vorhanden
// oder
// der Kommunikationspartner will eine Nachricht haben.

{ while (ocount < min(owindow, olength)
        || ireply == 1) {

    f = new packet;

    // Ich habe meine Pflicht getan und ein Paket gesendet.
    ireply = 0;

    // Die eigene Sicht der Phase und meine aktuelle
    // Sequenznummer mitgeben.
    f->dphase = ophase;
    f->dseqno = (oseqno + ocount) % m;

    // Wenn vorhanden und wenn möglich,
    // gleich ein Datenpaket mit übertragen.
    if (ocount < min(owindow, olength)) {
      // A packet should be included in the frame.

        // Es handelt sich um ein Datenpakete
      f->dpacket = 1;

        // Das nächste anstehende Paket wird übertragen und
        // Anzahl der noch nicht quittierten Pakete erhöhen
      f->data = oqueue(ocount++);
    }
```

```
    // Es gibt Pakete in der Warteschlange
    if (olength > 0) {
      // Not all output packets have as yet been
      // accepted.

        // Bestätigung oder eine Kontrollnachricht zum
        // Abgleich des Zustands anfordern.
      f->dreply = 1;
    }

    // Die eigene Sicht der Eingangsphase und der
    // Eingangssequenznummer mitteilen
    f->rphase = iphase;
    f->rseqno = iseqno;

    // Meine eigene aktuelle Fenstergröße für zu
    // empfangende Pakete mitteilen
    f->rwindow = iwindow;

    // Paket auf die Leitung ...
    send(f);
  }
}
```