

- Der *input_handler* wird für jedes eingehende Paket aufgerufen.
- Da Pakete fragmentiert sein können, sind dies möglicherweise Bruchstücke einer Anfrage oder auch Teile mehrerer Anfragen.
- Entsprechend muss die Eingabe wieder gepuffert und zerlegt werden, da normalerweise eine Reaktion erst bei einer vollständig übermittelten Anfrage erfolgen sollte.
- Eine ereignisgesteuerte Behandlung wäre daher aus Anwendungssicht leichter zu programmieren, wenn sie auf vollständigen Anfragen beruhen würde.
- Die Erkennung einer vollständigen Anfrage ist im allgemeinen Fall nicht ganz trivial zu spezifizieren. Im folgenden wird eine Lösung auf Basis regulärer Ausdrücke vorgestellt, die für textbasierte Protokolle gut geeignet ist.

mpx_session.h

```
typedef void (*mpx_handler)(session* s);

int mpx_session_scan(session* s, ...);
int mpx_session_printf(session* s, const char* restrict format, ...);
void close_session(session* s);

void run_mpx_service(hostport* hp, const char* regexp,
    mpx_handler ohandler, mpx_handler rhandler, mpx_handler hhandler,
    void* global_handle);
```

- *run_mpx_service* erhält einen regulären Ausdruck, der eine Anfrage spezifiziert.
- Dieser reguläre Ausdruck darf mit Hilfe runder Klammern beliebig viele Elemente der Anfrage herausgreifen – analog zu *inbuf_scan*.
- Der *rhandler* (*request handler*) wird dann für jede vollständig vorliegende Anfrage aufgerufen und kann dann mit *mpx_session_scan* die herausgegriffenen Elemente in *stralloc*-Objekte hineinkopieren lassen.

`mxprequest.h`

```
#define MXP_REQUEST_RE "([a-z]+) (.*)\r?\n"
```

`mutexd.c`

```
run_mpx_service(&hp, MXP_REQUEST_RE,  
               mpx_session_open, mpx_session_read, mpx_session_hangup,  
               locks);
```

- Beim Aufruf von `run_mpx_service` wird der reguläre Ausdruck zum Erkennen einer Anfrage mitgegeben.

mxpession.c

```
void mxp_session_read(session* s) {
    struct mxp_session* ms = s->handle; assert(ms);
    if (!read_mxp_request(s, &ms->request)) {
        close_session(s); return;
    }
    /* ... process request and generate response ... */
    if (!write_mxp_response(s, &ms->response)) {
        close_session(s);
    }
}
```

- Der Behandler *mxp_session_read* wird jetzt nur aufgerufen, wenn eine vollständige Anfrage vorliegt. Entsprechend sollte *read_mxp_request* eine Anfrage einlesen können.

mxprequest.c

```
bool read_mxp_request(session* s, mxp_request* request) {
    return
        mpx_session_scan(s, &request->keyword, &request->parameter) == 2;
}
```

mxresponse.c

```
/* write one (possibly partial) response to */
bool write_mxp_response(session* s, mxp_response* response) {
    return mpx_session_printf(s, "%c%.*s\r\n", response->status,
        response->message.len, response->message.s) > 0;
}
```

- Die Einlese-Operation für Anfragen und die Ausgabe-Operation für Antworten verwenden hier die entsprechenden Funktionen aus *mpx_session.h*