This document outlines requirements for the Mero RPC adaptive transmission (AT) feature.

AT provides an interface to attach memory buffers to an rpc item on the sender side. AT implementation transfers the contents of the buffer to the receiver either

- *inline*, *i.e.*, by including the contents of the buffer into the rpc item body, or
- *inbulk*, by constructing a network buffer representing the memory buffer, transmitting the network buffer descriptor in the rpc item and using bulk (RDMA) to transfer the contents of the buffer.

Independently of the transfer method selected, the receiver uses the same interface to access the contents of the buffer.

Requirements

- re-use m0_rpc_bulk for inbulk transfer mode
- the decision to use inline *vs.* inbulk is based on
    - buffer alignment. Only page-aligned buffers are suitable for inbulk,
    - buffer size and total rpc size. Rpc initialisation takes a new parameter that determines the cutoff size after which inbulk is used.
- receiver side is fom-oriented, *i.e.*, can be easily used within a tick function:

```
foo_tick(fom) {
        switch (fom_phase(fom)) {
        ...
        case PHASE_X:
                ...
                /*
                 * Start loading the buffer.
                 *
                 * If the buffer is passed inline, this returns
                 * FSO_AGAIN. If the buffer is passed inbulk, this
                 * initiates RDMA, arranges for the fom wakeup and
                 * returns FSO_WAIT.
                 *
                 * In any case the fom moves to PHASE_Y.
                 */
                return m0_rpc_at_load(&fop->at_buf, fom, PHASE_Y);
        case PHASE_Y:
```

```
                struct m0_buf buf;
                /*
                 * Returns loaded buffer.
                 *
                 * If inbulk transfer failed, returns error.
                 */
                rc = m0_rpc_at_get(&fop->at_buf, &buf);
                ...
        }
}
```

- a sender has an interface to add buffers (m0_buf) to an AT buffer, *e.g.*,

```
struct m0_rpc_at_buf {
        uint32_t ab_type; /* inline or inbulk */
        ...
} M0_XCA_UNION;

void m0_rpc_at_init(struct m0_rpc_at_buf *ab);
void m0_rpc_at_fini(struct m0_rpc_at_buf *ab);
/**
 * Called once for an AT buffer. If this returns success, the AT buffer
 * can be serialised as part of rpc item.
 */
int  m0_rpc_at_add(struct m0_rpc_at_buf *ab, const struct m0_buf *buf);
```