Особенности реализации библиотеки MPI на основе MPICH для сети Ангара

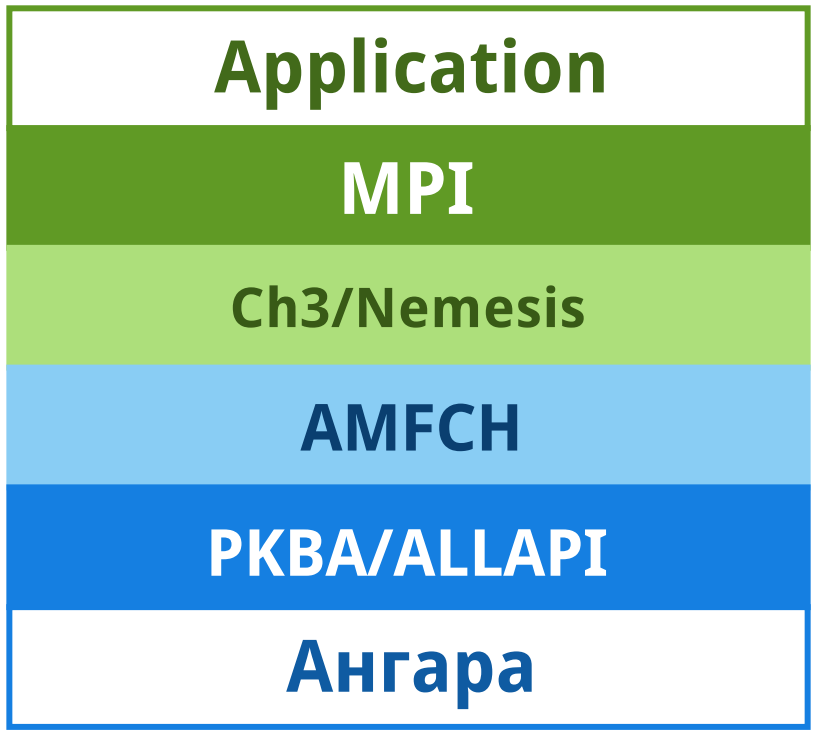
Поляков Д. А.

АО «НИЦЭВТ»

11 декабря 2017

Описание

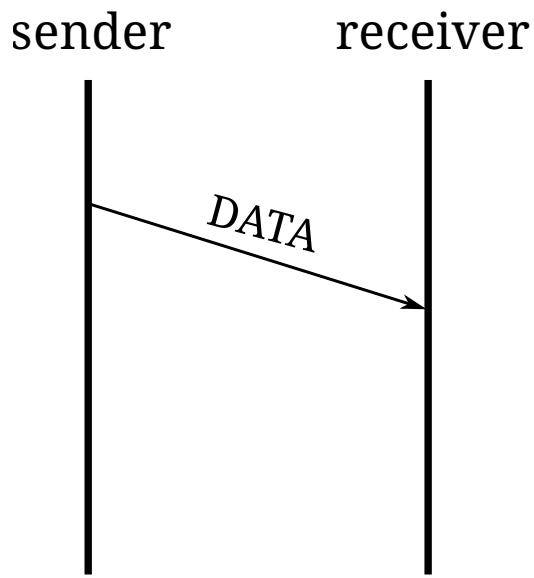
Поддержка сети Ангара в MPICH реализуется backend-модулем amfch (Angara Messaging Framework Channel), использующим стандартный интерфейс Nemesis Channel.



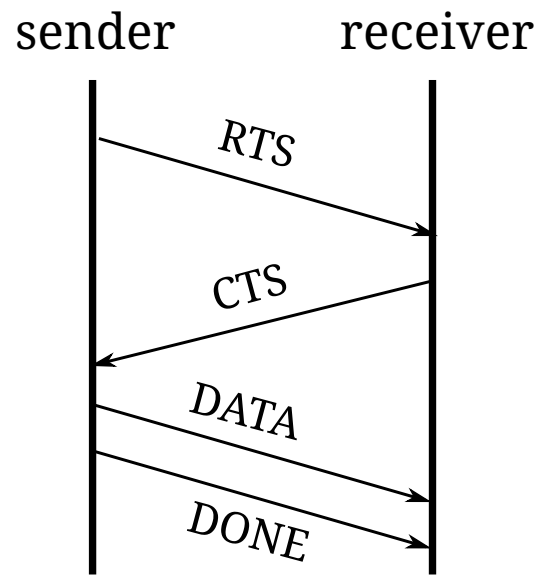
Описание

В зависимости от размера передаваемого сообщения в MPICH может использоваться один из двух протоколов:

- Eager (для передачи коротких сообщений);
- Rendezvous (для передачи длинных сообщений).



eager protocol



rendezvous protocol

Используемое API библиотеки PKBA

```
int pkba_init(bool console_mode, int access_mode);
void pkba_finalize(void);

u32 pkba_my_pe(void);

poffset_t pkba_procmem_alloc(size_t nwords); /* DMA mem */

void pkba_put_vc(u32 vch, const u64 *data,
    poffset_t poffset, u64 size, u32 pe);
void pkba_putoneword_vc(u32 vch, u64 value,
    poffset_t poffset, u32 pe);
void pkba_slget(poffset_t wrpoffset, poffset_t rdpoffset,
    u64 size, u32 from_pe, u32 to_pe);/* rendezvous only */
```

struct MPID_nem_netmod_funcs

```
MPID_nem_netmod_funcs_t MPIDI_nem_angara_funcs = {
    MPID_nem_angara_init,
    MPID_nem_angara_finalize,
    MPID_nem_angara_poll,
    MPID_nem_angara_get_business_card,
    MPID_nem_angara_connect_to_root,
    MPID_nem_angara_vc_init,
    MPID_nem_angara_vc_destroy,
    MPID_nem_angara_vc_terminate
}
```

Функции отправки сообщений

```
int iSendContig(MPIDI_VC_t *vc, MPID_Request *sreq,
    void *hdr, MPIDI_msg_sz_t hdr_sz, void *data,
    MPIDI_msg_sz_t data_sz);

int iStartContigMsg(MPIDI_VC_t *vc, void *hdr,
    MPIDI_msg_sz_t hdr_sz, void *data,
    MPIDI_msg_sz_t data_sz, MPID_Request **sreq_ptr);

int SendNoncontig(MPIDI_VC_t *vc, MPID_Request *sreq,
     void *hdr, MPIDI_msg_sz_t hdr_sz);
```

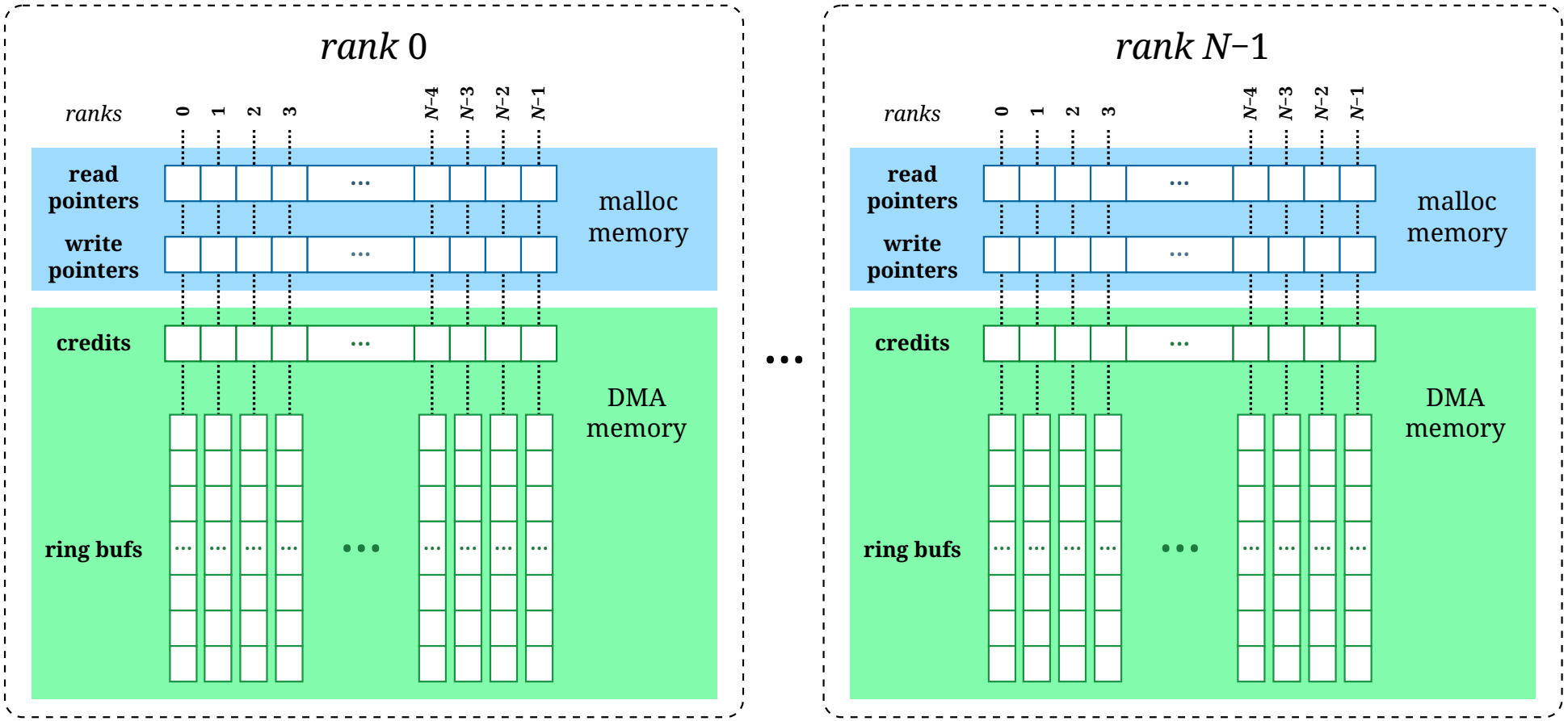
```
sizeof(*sreq) = 864 bytes, sizeof(*hdr) = 40 bytes.
```

struct MPIDI_VC_t

```
struct MPIDI_VC_t { /* VC - virtual connection */
    ...
    int pg_rank;
    ...
    int eager_max_msg_sz;
    ...
    sendNoncontig_fn;
    iStartContigMsg
    iSendContig
    ...
    char padding[MPID_NEM_VC_NETMOD_AREA_LEN /*128*/];
}
```

sizeof(struct MPIDI_VC_t) = 456 bytes.

Инициализация. Память.



Обмен сообщениями. Отправка.

```
int MPI_Send(const void *buf, int count,
    MPI_Datatype datatype, int dest,
    int tag, MPI_Comm comm)
{
    MPID_Request *request_ptr = NULL;
    ...
    MPID_Send(buf, count, datatype, dest, tag, comm_ptr,
        MPID_CONTEXT_INTRA_PT2PT, &request_ptr);

    while (!MPID_Request_is_complete(request_ptr))
        MPID_Progress_wait(&progress_state);
    ...
    return MPI_SUCCESS;
}
```

Обмен сообщениями. Отправка.

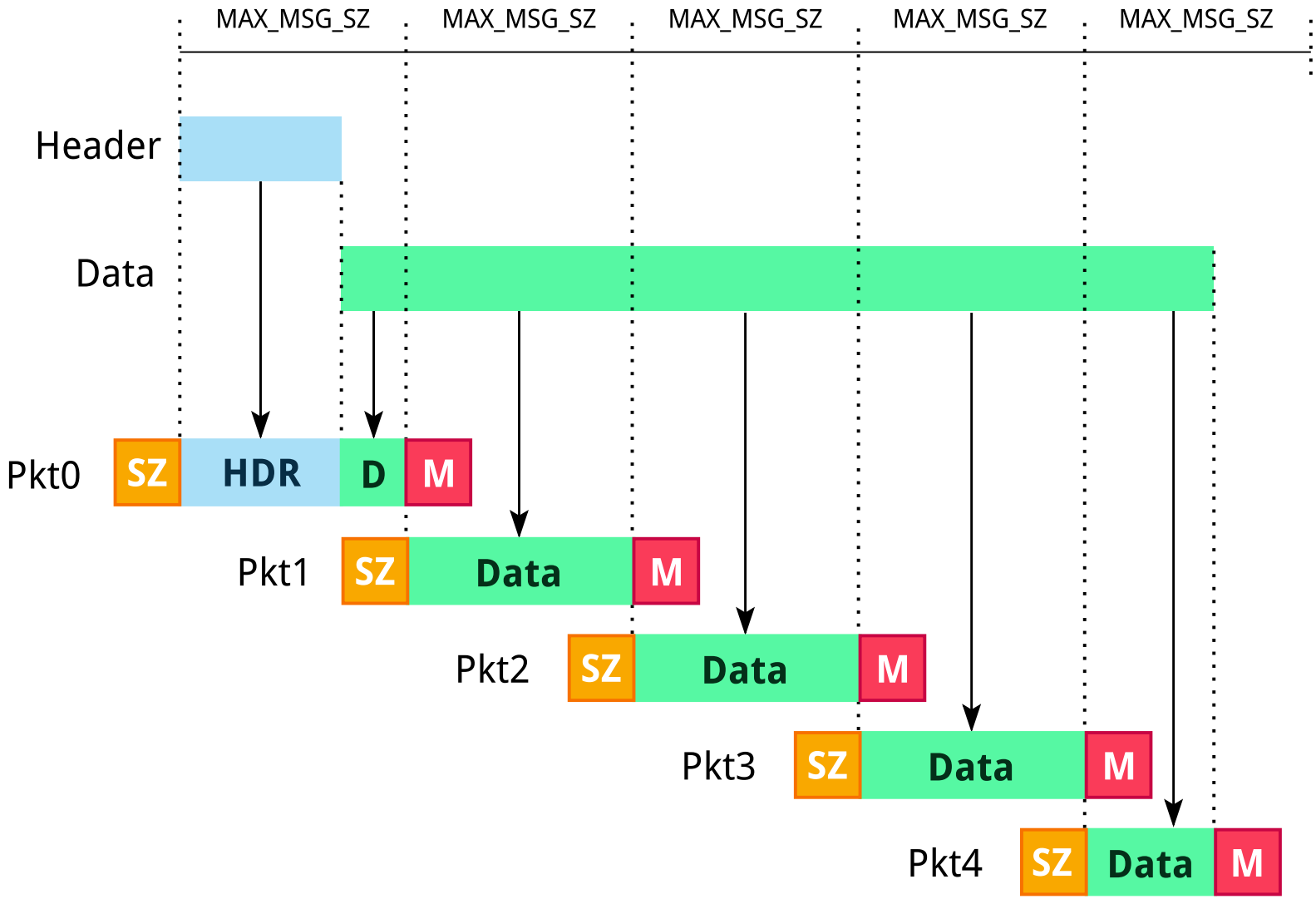
```
int MPID_Send(...) {
    if (dt_contig && data_sz <= EAGER_SHORT_SIZE /*16*/) {
        MPIDI_CH3_EagerContigShortSend(...);
    }
    else if (data_sz + eager_hdr <= vc->eager_max_msg_sz) {
        if (dt_contig) {
            MPIDI_CH3_EagerContigSend(...);
        } else {
            MPIDI_CH3_EagerNoncontigSend(...);
        }
    } else {
        vc->rndvSend_fn(...);
    }
}
```

Обмен сообщениями. Отправка.

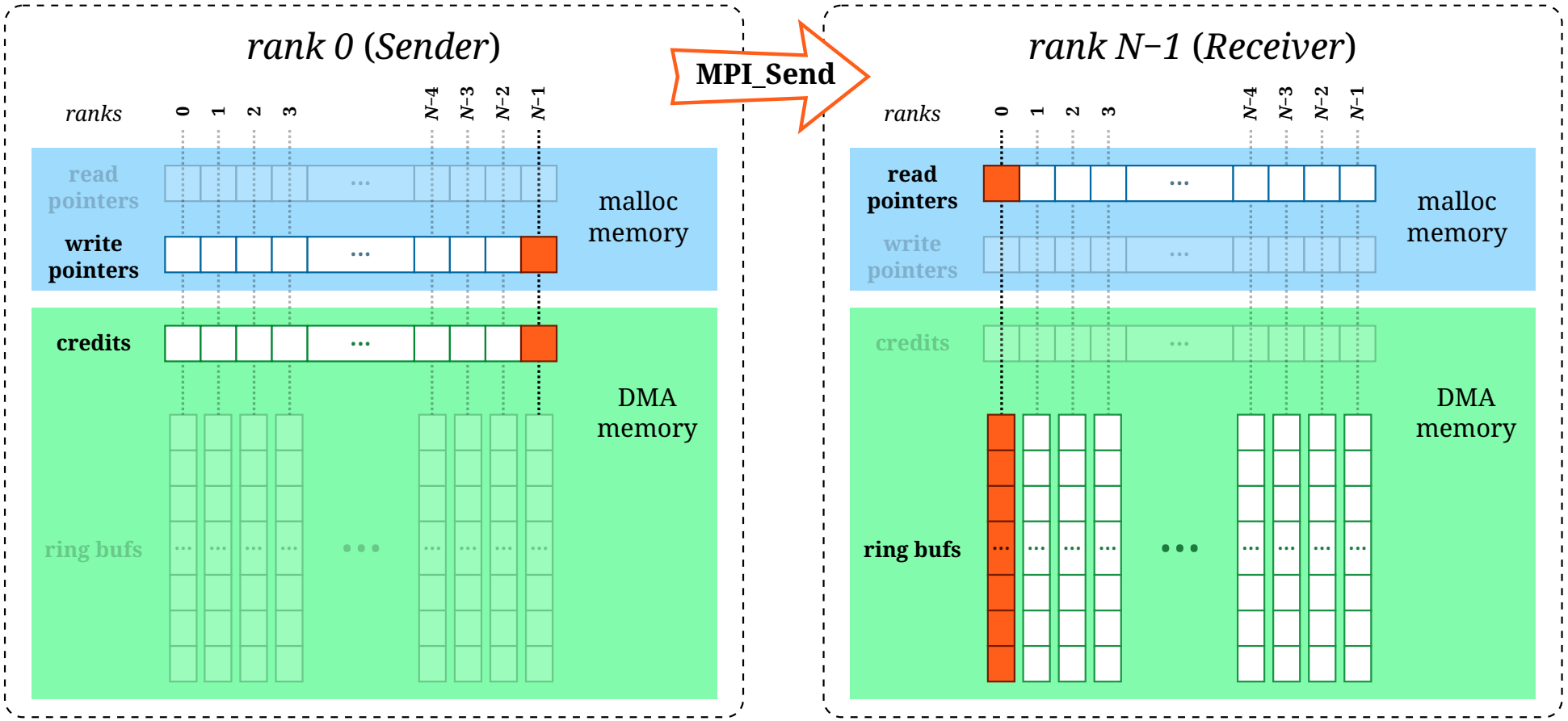
```
typedef union {
    MPIDI_Message_match_parts_t parts;
    MPIR_Upint whole;    /* unsigned long */
} MPIDI_Message_match;

struct MPIDI_CH3_Pkt_eagershort_send {
    MPIDI_CH3_Pkt_type_t type;    /* enum */
    MPIDI_Message_match match;    /* tag, rank, context */
    MPIDI_msg_sz_t data_sz;
    int data[MPIDI_EAGER_SHORT_INTS];
}
```

Обмен сообщениями. Отправка.



Обмен сообщениями. Отправка.

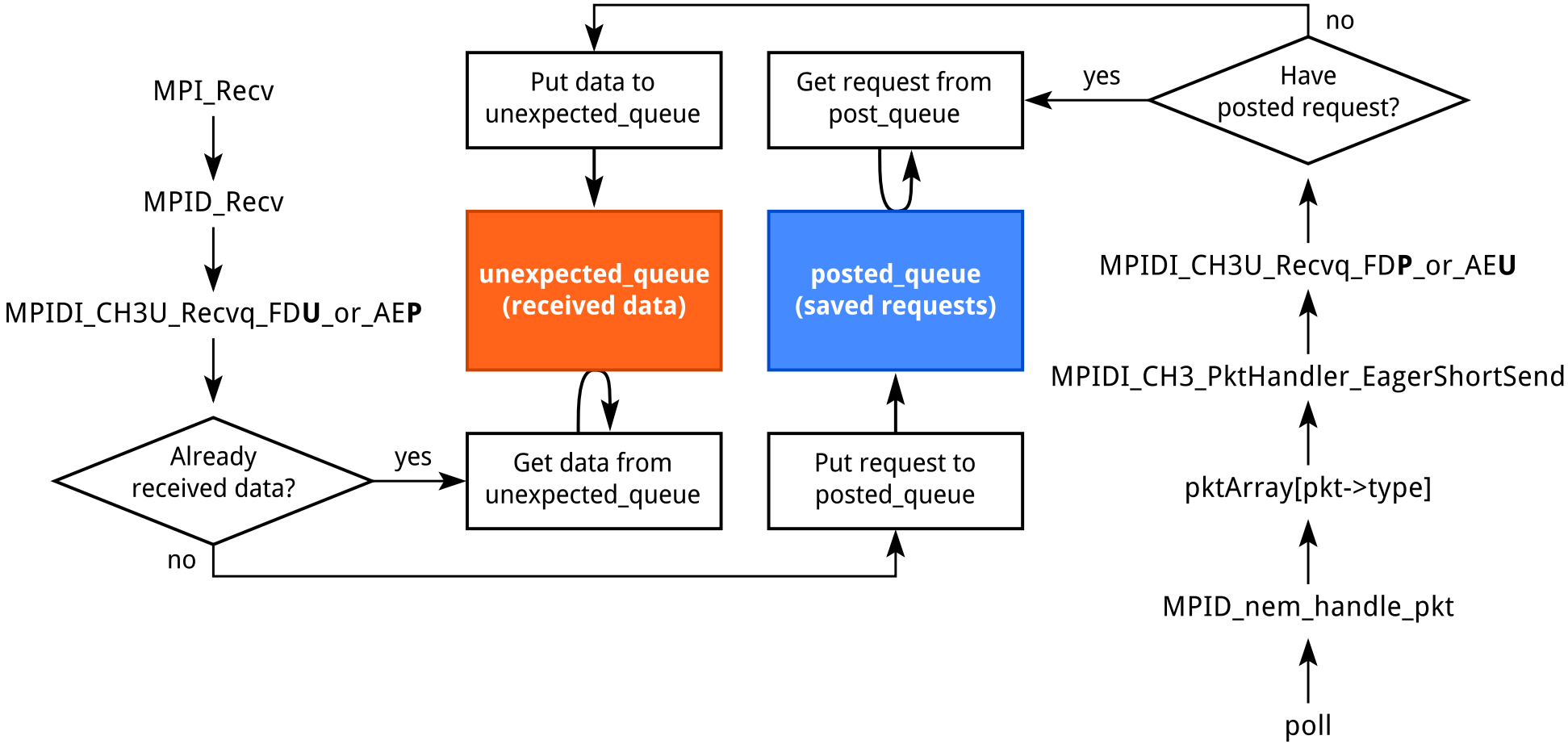


Обмен сообщениями. Функция poll().

Функция poll() используется для:

- Приёма сообщений (`MPID_nem_handle_pkt()`);
- Передачи сообщений, которые были поставлены в очередь отправки;
- Приёма/передачи сообщений по протоколу rendezvous.

Обмен сообщениями. Приём.



Обмен сообщениями. Приём.

```
int MPI_Recv(void *buf, int count,
    MPI_Datatype datatype, int source, int tag,
    MPI_Comm comm, MPI_Status *status)
{
    MPID_Request * request_ptr = NULL;
    ...
    MPID_Recv(buf, count, datatype, source, tag, comm_ptr,
        MPID_CONTEXT_INTRA_PT2PT, status, &request_ptr);

    while (!MPID_Request_is_complete(request_ptr))
        MPID_Progress_wait(&progress_state);
    ...
    return MPI_SUCCESS;
}
```

Обмен сообщениями. Оптимизация.

- Основная идея — проверять не все буферы, а только те, для которых ожидается получение сообщения;
- Оптимизация barrier, allreduce.