MTCP : TRANSPORT LAYER
SUPPORT FOR HIGHLY AVAILABLE NETWORK SERVICES

Kiran Srinivasan
New Brunswick, New Jersey
October, 2001
Outline

• Introduction
• Implementation
• Experiment
• conclusion
Introduction

• What is migration?
  – change the connection by binding App. State, socket state, TCP state etc.

• When do migration?
  – Server overload, congestion occur, DoS attack etc.
Introduction

- **Migration issues**
  - Triggers and initiators
    - Client, S1 or S2
  - Export/import state snapshot
    - App. state, socket state
  - State synchronization
    - Because of App. and protocol are asymmetric
  - Buffer synchronization
```
while((s = accept(ssock)) != -1)
{
    if( import_state(s, &state))
        num = state
    else
        num = 0
    recv (s, &buf)
    state = ++num
    export_state (s, state)
    recv (s, &buf)
    state = ++num
    export_state (s, state)
    ...
}
```
State Synchronization
Buffer Synchronization
Implementation

Lazy migration scheme
Implementation

typedef struct
{
    u_int32_t server_ip_address;    /* the server’s ip address that it
                                    * uses to interact with the client */
    u_int32_t control_ip_address;  /* Control connection’s IP address */
} ctrl_ip_mapping;

#define MAX_CONTROL_IPS MAX_IP_ADDRS
struct ctrl_mappings_info
{
    ctrl_ip_mapping ctrl_mappings[MAX_CONTROL_IPS];
    /* the mappings */
    char num_maps; /* the number of mappings */
};
Implementation

- **Read_log_mb**
  - A chain of mbufs that stores all complete mbufs read

- **So_read_log_len**
  - Read_log_mb mbuf chain size

- **So_read_log_offset**
  - First mbuf present in the read buffer
Implementation

- Additional servers information transfer via TCP option
  - Client copy the info. to self - extra_ip structure
Implementation

- When two flag is setup
  - Server has so_migration_supported
  - Client has so_can_migrate

- Client send SYN + TCP_MIGRATED_CONN option to S2
  - The connection ID of C and S1 Rcv_nxt in original connection

- Client socket enter SS_ISMIGRATING
  - Don’t send any data to S1
When S2 receives a client request, it initiates the migration process. The diagram shows that S2 sends a `SYN` message and a `MIGRATE_REQUEST` with parameters `{S1_ip, S1_port, C_ip, C_port}`. The migration process involves updating the TCP control block, control mapping table, hash table, and control socket.
Implementation

• **State request**
  - 4-tuple connection ID that is migrating
  - 4-tuple connection ID for new server
  - `Rcv_nxt` from client to `s2`

• **State_reject**
  - An integer representing the reason
    • `INVALID STATE`
Implementation

- State reply
  - Application-level state length

```c
struct tcp_conn_state
{
  struct mbuf
  int
  tcp_seg
  tcp_seg
  struct socket_snapshot
  struct tcp_snapshot
} t_conn_state

struct tcp_snapshot
{
  short so_state;
  short so_options;
  short so_linger;
};
```
Implementation

- Write buffer state length
  - $\text{Snd\_max\_seen} = \max(\text{rcv\_nxt}, \text{syn\_una})$
Implementation

- Receive buffer state length
  - \( \text{Rcv\_state\_length} = (\text{last\_ack\_sent} - \text{rcv\_seq\_snapshot}) \)
Implementation

• State reply packet header
  - A tag indicating the packet is state reply
  - The application state length
  - Write buffer state length
  - Read buffer length
  - Rcv_seq_snapshot and last_ack_sent
  - Snd_bytes_discard
  - Snd_max_seen
  - T_snapshot
  - So_snapshot
Implementation

- Header and logged data are organized as a chain of mbufs is sent over the control connection.
Implementation

• S2 send ACK
  - S1 enter passive close state
  - When S1 receive implicit ACK, closed

• S2 send SYN, rcv_nxt(S1) to client
  - Client terminate the SS_ISMIGRATING state
  - Client resume the connection
Experiment

- Celeron 450MHz processor
- 256MB RAM
- 100Mbps Ethernet links
Experiment

When server send data, client do migration every time.
Conclusion

- MTCP allows dynamic connection migration at any point of time during the session
- Basic assumption: per-connection state is represented as a chunk of memory
  - Recursive migration