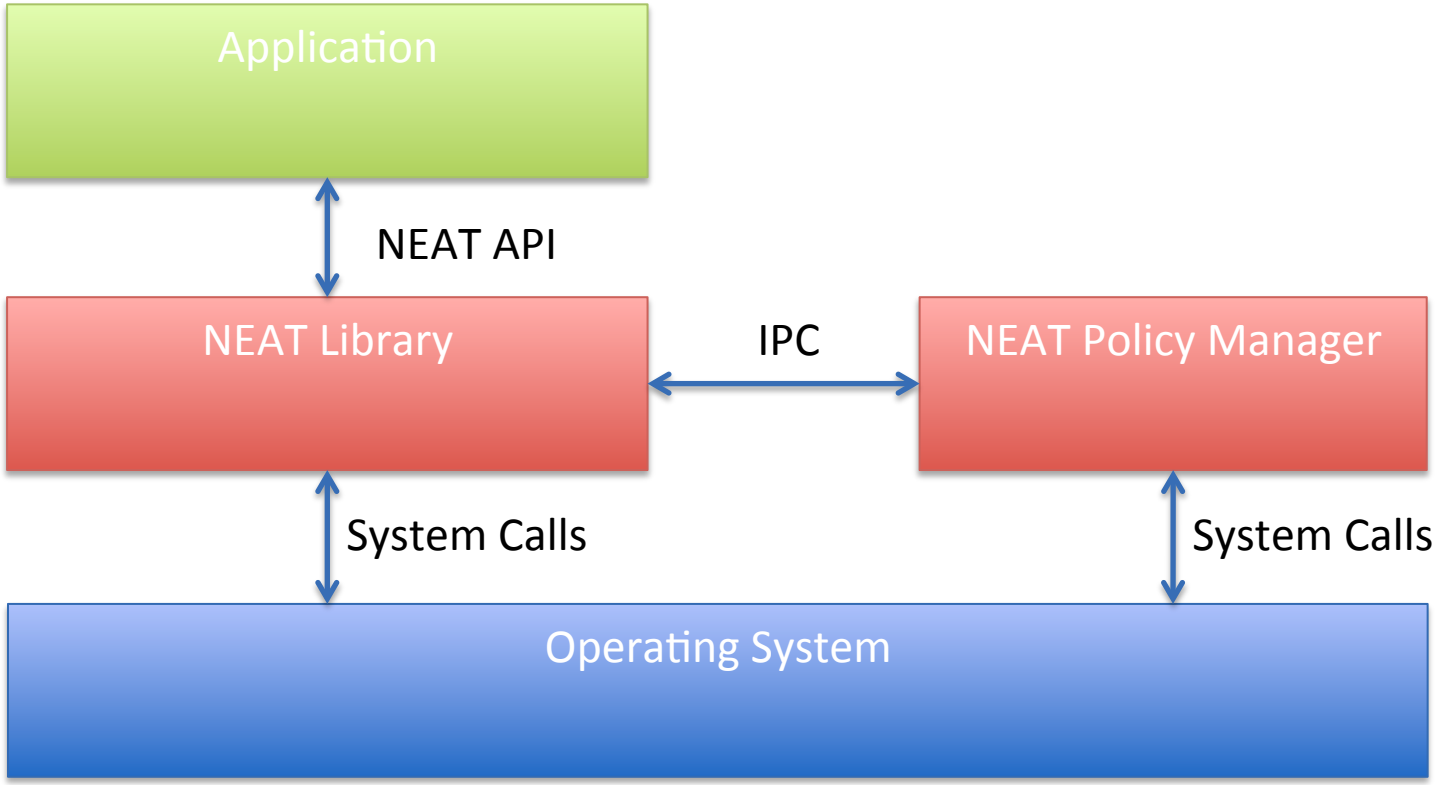
neat

The NEAT API – A Case Study

Michael Tüxen

Horizon 2020
European Union funding
for Research & Innovation

Incomplete High Level View of NEAT



Key Features

- Event based, using callbacks
- OS-independent networking API
- Programming language independent
- Protocol independent API
- Selection of protocols and parameterization based on
  - Configured policies
  - Tested capabilities
  - Known and learned capabilities

A Simple Client using the NEAT API (1)

```
static char *properties =
"{\"transport\": {\"value\": \"reliable\", \"precedence\": 2}}";

int main(void) {
    struct neat_ctx *ctx;
    struct neat_flow *flow;
    struct neat_flow_operations ops;

    ctx  = neat_init_ctx();
    flow = neat_new_flow(ctx);
    memset(&ops, 0, sizeof(ops));
    ops.on_connected = on_connected;
    neat_set_operations(ctx, flow, &ops);
    neat_set_property(ctx, flow, properties);
    neat_open(ctx, flow, "bsd10.fh-muenster.de", 5000, NULL, 0);
    neat_start_event_loop(ctx, NEAT_RUN_DEFAULT);
    neat_free_ctx(ctx);
    return EXIT_SUCCESS;
}
```

A Simple Client using the NEAT API (2)

```
static neat_error_code on_connected(struct neat_flow_operations *ops) {
    ops->on_writable    = on_writable;
    ops->on_all_written = on_all_written;
    neat_set_operations(ops->ctx, ops->flow, ops);
    return NEAT_OK;
}

static neat_error_code on_writable(struct neat_flow_operations *ops) {
    neat_write(ops->ctx, ops->flow, "Hi!", 3, NULL, 0);
    return NEAT_OK;
}

static neat_error_code on_all_written(struct neat_flow_operations *ops) {
    ops->on_readable = on_readable;
    ops->on_writable = NULL;
    neat_set_operations(ops->ctx, ops->flow, ops);
    return NEAT_OK;
}
```

A Simple Client using the NEAT API (3)

```
static neat_error_code on_readable(struct neat_flow_operations *ops) {
    uint32_t bytes_read = 0;
    char buffer[32];

    if (neat_read(ops->ctx, ops->flow, buffer, 31,
                  &bytes_read, NULL, 0) == NEAT_OK) {
        buffer[bytes_read] = 0;
        fprintf(stdout, "Read %u bytes:\n%s", bytes_read, buffer);
    }
    neat_close(ops->ctx, ops->flow);
    neat_stop_event_loop(ops->ctx);

    return NEAT_OK;
}
```