ClassAttributes
noch in die init
Funktionen packen?

«application start»
main
{abstract}

-main()
-parse_config_arguments()

Service Manager

Service Manager :: ServiceManagerCore

- ADJACENCY_LIST_FILE_NAME_PATH: String = "../adjacency_list.json"
- MAX_NUM_NODES: Int = 60
- SERVER_PORT: Int = 6001
- SERVICE_FILE_NAME_PATH: String = "./service.py"
- CONNECTION_CHECK_TIME: Int = 10
- CPU_RAM_CHECK_TIME: Int = 1
- CPU_THRESHOLD: Float = 20.0
- RAM_THRESHOLD: Float = 15.0
- TESTING_FLAG: Boolean = False
- MIGRATION_FLAG: Boolean = True

- start_service_event: Event = False
- migrate_service_event: Event = False
- duplicate_service_event: Event = False
- unreachable_hosts: Int[0..*]

~ network_router: NetworkRouting
- migration_checker: MigrationChecker
- network_sniffer: NetworkSniffer
~ service_handler: ServiceHandler
- service_transporter: ServiceTransporter

+ init(ClassAttributes): ServiceManagerCore «constructor»
+ cb_service_transporter_service_received(): Int, String
+ cb_service_handler_new_service_ports_found(service_pid: Int,
ports: Int[0..*])
+ cb_network_sniffer_new_packet(packet_size: Int)
+ cb_network_sniffer_new_packet_for_service(source_ip_addr: String,
dest_ip_addr: String,
packet_size: Int,
is_incoming_packet: Boolean)
+ cb_migration_checker_send_service(duplicate_service, reason)
+ run()

Service Manager :: ServiceHandler

- testing_flag: Boolean
- own_node_id: Int
- service_file_name_path: String
- cb_new_service_ports_found: Function

- service: subprocess.Popen = None
- service_ports: Int[0..*] = None
- service_status: Tuple(Int, String)
- service_status_lock: threading.Lock
- open_ports_check: Utils.RepeatedTimer

+ init(ClassAttributes): ServiceHandler «constructor»
+ set_service_status(status: Int, error: String) «setter>
+ get_service_status(): Tuple(Int, String) «getter»
+ delete_service_and_reset_status(): Boolean
- broadcast_service_status_to_clients(service_name: String,
event: String)
- seperate_ipv4_and_port(combined_ip_port: String): String, Int
+ get_open_ports_of_service()
+ start_service(): Tuple(Int, String)
+ stop_service(): Boolean

Service Manager :: ServiceTransporter

- service_handler: ServiceHandler
- service_file_name_path: String
- server_port: Int
- cb_service_received: Function

- server_socket: socket.socket
- receive_service_lock: threading.Lock
- server_thread: threading.Thread

+ init(ClassAttributes): ServiceTransporter «constructor»
+ run()
+ receive_service(conn: socket.socket)
+ send_service(host: String, file_path: String)

Utils

Utils :: MigrationChecker

- migration_flag: Boolean
- testing_flag: Boolean
- network_router: NetworkRouter
- cb_send_service: Function
- cpu_ram_check_time: Int
- cpu_threshold: Float
- ram_threshold: Float

- check_connections_timer: RepeatedTimer
- connection_check_lock: threading.Lock
- recently_connected_nodes_counter: Dict
- recently_connected_nodes_total_counter: Int
- best_new_choosen_node: Int

- check_cpu_ram_timers: List[0.*]
- cpu_ram_check_lock: threading.Lock
- recent_cpu_ram_usage: Dict

+ init(ClassAttributes): MigrationChecker «constructor»
+ start_forever(): Boolean
+ cancel_migration_check()
+ check_recent_cpu_and_ram_usage(pid: Int)
+ start_recent_cpu_and_ram_usage_timer(pid: Int)
+ get_best_new_choose_node(): Int
+ add_new_connection(node_id: Int, packet_size: Int)
+ calculate_avg_cpu_and_ram_out_of_recent_usage(): Float, Float
+ check_recent_connections_for_best_server()

Utils :: NetworkRouting

- testing_flag: Boolean
- own_hostname: Int
- adjacency_list: List[0..*]
- total_num_hosts: Int
- nodes_connection_time_table: Int

+ init(ClassAttributes): NetworkRouting «constructor»
+ get_own_hostname(): Int «getter»
+ startup_wlan_interfaces()
- dijkstra(initial: Int): List[1..*], List[1..*]
- shortest_path(origin: Int, destination: Int): Float, List[0..*]
+ add_all_network_routes()
- calculate_nodes_connection_time_table()
+ calculcate_central_node_from_recent_connections(
recently_connected_nodes_counter: Dict,
recently_connected_nodes_total_counter: Int)

«utility»
Utils :: NetworkFunctions

#Operation(i: int): int
- recvall(sock: socket.socket, n: Int, timeout: Int): String
+ send_packed(sock: socket.socket, msg: String)
+ recv_packed)sock: socket.socket, timeout: Int): String
+ translate_ip_addr_to_node_id(ip_addr: String): String
+ translate_node_id_to_ip_addr(node_id: String): String
- get_all_interfaces(): List[0..*]
+ get_wireless_interfaces(): List[0..3]

Utils :: NetworkPacket

+ packet: String

+ total_size: Int

+ source_mac: String = None
+ dest_mac: String = None
+ ether_type: String = None

+ ip_version: String = None
+ ihl: String = None
+ ttl: String = None
+ protocol: Int = None
+ source_ip_address: String = None
+ dest_ip_address: String = None

+ source_port: Int = None
+ dest_port: Int = None
+ seq_number: String = None
+ ack_number: String = None
+ doff_reserved: String = None
+ data: String = None
+ data_size: String = None

+ icmp_type: String = None
+ code: String = None
+ checksum: String = None

+ udp_length: String = None

+ init(packet_string: String): NetworkPacket «constructor»
- eth_addr(a: List[6]): String
- extract_ethernet_header()
- extract_ipv4_header()
- extract_tcp_header()
- extract_icmp_header()
- extract_udp_header()

Utils :: NetworkSniffer

- testing_flag: Boolean
- own_node_id: int
- ports: List[0..*]
- stopped_event: threading.Event
- cb_new_packet: Function
- cb_new_packet_for_service: Function
- network_sniffer_thread: threading.Thread
- tcpdump_hex_row_regex: RegularExpression
- tcpdump_hex_content_regex: RegularExpression
- wireless_interfaces: List[0..3]

+ init(ClassAttributes): NetworkSniffer «constructor»
+ run()
+ cancel_sniffing()
+ set_sniffing_ports()

Utils :: RepeatedTimer

- stopped_event: threading.Event
- seconds: Int
- cb_function: Function
- args: Tuple[0..*]

+ init(seconds: Int,
cb_function: Function,
args: Tuple[0..]): RepeatedTimer «constructor»
+ run()
+ cancel