src Documentation

Release

Author

December 11, 2015

Contents

Contents:

CHAPTER 1

blackwidow package

1.1 Subpackages

1.1.1 blackwidow.graph package

Submodules

blackwidow.graph.graph_rate module

class blackwidow.graph.graph_rate.**CsvGrapher**(*bw*)
Bases: object

Graphs the .csv files.

Parameters **bw** : *BlackWidow*

BlackWidow simulation object containing simulation settings.

Attributes

bw	(*BlackWidow*) BlackWidow simulation object containing simulation settings.
data_dir	(string) Data directory containing .csv files.
log_file	(string) Base file name for the log file indicating case number.
smooth_factor	(int) Number of data points to average for rates.
max_capacity	(float) Maximum capacity for link rates to throw out outliers.

Methods

graph(sim_time)	Graph all desired rates based on the csv files.

class blackwidow.graph.graph_rate.**GraphSettings**(*data_dir*, *log_file*, *sim_time*)
Bases: object

Emulates bw object if run from script.

1.1.2 blackwidow.network package

Submodules

blackwidow.network.device module

class blackwidow.network.device.**Device**(*net_addr*)
Bases: object

Super class for the *Host* and *Router* classes.

Parameters **net_addr** : string

A unique id for the device in the network.

Attributes

network_id	(string) A unique id of the device in the network.
links	(list) A list of links that the device is connected to.

Methods

add_link(link)	Adds the specified *Link* object to *links*.
delete_link(link)	Remotes the specified *Link* object from *links*.

add_link(*link*)
Add link to list of links.

Parameters **link** : *Link*

The link to add to the device.

delete_link(*link*)
Remove link from list of links.

Parameters **link** : *Link*

The link to remove from the device.

send(*packet*)
Virtual method for sending device packets.

Parameters **packet** : *Packet*

Packet to send.

blackwidow.network.event module

class blackwidow.network.event.**Event**(*type*, *src_id*, *f*, ***kwargs*)
Event object to run.

This object contains a function to run with any arguments. This is used by other objects to run specific functions at some time.

Parameters **type** : string

A message specifying the type of the event.

src_id : string

The id of the source object creating the event.

f : func

The function to run.

kwargs : dict

Keyword arguments to provide to *f*.

Notes

The event is initialized with a id. The *Event* class keeps a static id that is updated for each event to create a unique id.

Attributes

id	(string) The event id.
src_id	(string) The id of the object that created the link.
type	(string) The type of the event.

Methods

run()	Runs the event.

run()

Runs the event.

Calls the function *f* with keyword arguments *kwargs*.

blackwidow.network.fast_flow module

class blackwidow.network.fast_flow.**FastFlow**(*flow_id*, *source*, *destination*, *amount*, *env*, *time*, *bw*)

Bases: *blackwidow.network.flow.Flow*

Implements FAST TCP. Flows will trigger host behavior.

Parameters **flow_id** : string

A unique id for the flow.

source : *Device*

The source for the flow.

destination : *Device*

The destination for the flow.

amount : int

The amount of data to send in MB.

env : *Network*

The network that the flow belongs to.

time : float

The amount of time to wait before starting to send in ms.

bw : Blackwidow

The printer to print data to

Attributes

flow_id	(string) The flow id.
src	(*Device*) The source for the flow.
dest	(*Device*) The destination for the flow.
amount	(int) The amount of data left to send in MB.
env	(*Network*) The network that the flow belongs to.
flow_start	(float) The amount of time to wait before starting to send. Specified in ms.
pack_num	(int) The next pack_num to check to send.
cwnd	(float) Congestion window size.
ssthresh	(float) Slow start threshold
resend_time	(float) ms before packets are sent after an ack receival
min_RTT	(float) Minimum round trip time observed for this flow
last_RTT	(float) Last round trip time observed for this flow
SRTT	(float) Weighted average of round trip times biased towards recent RTT
RTTVAR	(float) Variance of round trip times
RTO	(float) Retransmission timeout in ms
packets_sent	(list) List of packets that have been sent but haven't had their ack received
packets_time_out	(list) List of packets that have exceeded timeout and need to be resent
acks_arrived	(set) Set of ack packets that have been received
done	(int) 0 if flow isn't finished; 1 if flow is finished. Used to avoid decrementing flow more than once.
send_rate	(Rate_Graph) Keeps track of the rate the flow is sending at and outputs to CSV file in real time.
receive_rate	(Rate_Graph) Keeps track of the rate the flow is receiving at and outputs to CSV file in real time.
alpha	(float) alpha in FAST TCP algorithm; alpha = 20 because link rates are between 10 Mbps and 1 Gbps.
gamma	(float) gamma in FAST TCP algorithm; smoothing factor for window size
total_num_pack	(int) total number of packets that need to be sent

Methods

send_packet()	Send a packet.

_reset_window()
This is called when a packet timeout occurs by the parent Flow class. Does nothing since FAST TCP automatically updates every 20 ms.

_respond_to_ack()
Overwrites parent Flow class' method because it shouldn't change window size.

_update_window()
Send a packet.

send_packet()
Send a packet. The difference between FastFlow's send_packet and Flow's send_packet is the ending behavior. FastFlow just keeps resending packets it hasn't received yet until it is done after it has sent all the packets once.

blackwidow.network.flow module

class blackwidow.network.flow.**Flow**(*flow_id*, *source*, *destination*, *amount*, *env*, *time*, *bw*)
Bases: object

Simple class for flows. Flows will trigger host behavior. Has slow start and congestion avoidance.

Parameters **flow_id** : string

A unique id for the flow.

source : *Device*

The source for the flow.

destination : *Device*

The destination for the flow.

amount : int

The amount of data to send in MB.

env : *Network*

The network that the flow belongs to.

time : float

The amount of time to wait before starting to send. Specified in ms.

bw : Blackwidow

The printer to print data to

Attributes

flow_id	(string) The flow id.
src	(*Device*) The source for the flow.
dest	(*Device*) The destination for the flow.
amount	(int) The amount of data left to send in MB.
env	(*Network*) The network that the flow belongs to.
flow_start	(float) The amount of time to wait before starting to send. Specified in ms.
pack_num	(int) The next pack_num to check to send.
cwnd	(float) Congestion window size.
ssthresh	(float) Slow start threshold
resend_time	(float) ms before packets are sent after an ack receival
min_RTT	(float) Minimum round trip time observed for this flow
last_RTT	(float) Last round trip time observed for this flow
SRTT	(float) Weighted average of round trip times biased towards recent RTT
RTTVAR	(float) Variance of round trip times
RTO	(float) Retransmission timeout in ms
packets_sent	(list) List of packets that have been sent but haven't had their ack received
packets_time_out	(list) List of packets that have exceeded timeout and need to be resent
acks_arrived	(set) Set of ack packets that have been received
done	(int) 0 if flow isn't finished; 1 if flow is finished Used to avoid decrementing flow more than once.
send_rate	(Rate_Graph) Keeps track of the rate the flow is sending at and outputs to CSV file in real time.
receive_rate	(Rate_Graph) Keeps track of the rate the flow is receiving at and outputs to CSV file in real time.

Methods

receive(packet)	Generate an ack or respond to bad packet.
send_packet()	Send a packet.

_reset_window()
Called when a packet timeout occurs. Sets ssthresh to max(2, cwnd/2) and cwnd to 1.

_respond_to_ack()
Update window size.

_send_ack(*packet*)
Creates ack for packet. Parameters ———- packet : *Packet*

The packet to be received.

_timeout(*pack_num*)
Generate an ack or respond to bad packet. Parameters ———- pack_num : ‘Packet‘number

The packet number of the packet to check for timeout.

_update_RTT(*packet*)
Update last RTT and min RTT and retransmission timeout. Parameters ———- packet : *Packet*

The packet that was received. Need this to get the timestamp

receive(*packet*)
Generate an ack or respond to bad packet. Parameters ————- packet : *Packet*

The packet to be received.

send_packet()
Send a packet.

blackwidow.network.host module

class blackwidow.network.host.**Host**(*host_id*)
Bases: *blackwidow.network.device.Device*

Simple class for hosts.

Hosts are mainly responsible for recording their time data. They don't trigger events in the simulation, but it will be useful to separate host data (end to end data). Flows will trigger host behavior.

Parameters **host_id** : string

A unique id for the host.

Attributes

network_id	(string) A unique id of the device in the network.
links	(list) A list of links that the host is connected to.
flows	(list) A list of flows that use the host.

Methods

add_flow(flow)	Adds receiving flow to host.
delete_flow(flow)	Delete flow from the host.
send(packet)	Sends a packet to a link.
receive(packet)	Receives a packet from a link.

add_flow(*flow*)
Add receiving flow to host.

Parameters **flow** : *Flow*

The flow to add to the host.

delete_flow(*flow*)
Delete flow from host.

Parameters **flow** : *Flow*

The flow to add to the host.

receive(*packet*)
Send packet to flow to process.

Parameters **packet** : *Packet*

The packet to be received.

send(*packet*)
Connects to a link.

Parameters **packet** : *Packet*

The packet to send.

blackwidow.network.link module

class `blackwidow.network.link.`**`Link`**(*id*, *device_a*, *device_b*, *delay*, *rate*, *capacity*, *env*, *bw*)

Bases: `object`

Simulates a link connected to two Devices in the network.

Represents a physical link in the network. In addition to simple send and receive, this class also handles the packet buffer for sending packets.

Parameters **id** : string

A unique id for the link.

device_a : *Device*

A *Device* to which the link is connected.

device_b : *Device*

A *Device* to which the link is connected.

delay : float

The propagation delay to send packets across the link. Specified in ms.

rate : float

The rate at which the link can send a packet. Specified in Mbps.

capacity : int

The capacity of the link buffer. Specified in KB.

env : *Network*

The network that the link belongs to.

bw : *Blackwidow*

The simulation object containing settings and data recording.

Attributes

id	(string) The link id.
device_a	(*Device*) One of the *Device* objects to which the link is connected.
device_b	(*Device*) One of the *Device* objects to which the link is connected.
delay	(float) The progapation delay in ms.
rate	(float) The rate at which the link can send a packet in bits per ms.
capacity	(int) The capacity of the link buffer in bits.
distance	(float) The distance of the link. Used for dynamic routing.

Methods

receive(packet, source_id)	Receives a packet from a *Device*.
measure_distance()	Measures the link distance.

_release()
Releases the packet being sent to the receiving *Device* after the packet has traversed the link.

Notes

This function dequeues the first packet in the buffer and begins sending it across the link. The packet is sent to its destination after delay time, where delay is the propagation delay of the link. Routing packets and acknowledgement packets are sent instantaneously to their destination without considering the propagation delay. This simplifies the network simulation.

_send()
Sends the first packet in the buffer across the link.

Notes

The packet begins to transmit across the link after size / rate time, where size is the packet size and rate is the rate of the link. This function then calls _release to send the packet to the receiving *Device*.

get_buffer_size()
Returns the buffer size in bits.

measure_distance()
Measure the link distance.

Sets the distance attribute of the link.

receive(*packet*, *source_id*)
Receives a packet from a *Device*.

This function takes as parameter a *Packet* and a device id. Packets are either enqueued in the link buffer if the link buffer is not full or are dropped.

Parameters **packet** : *Packet*

The packet received by the link.

source_id : string

The id of the *Device* object sending the packet.

blackwidow.network.network module

class blackwidow.network.network.**Network**(*bw*)
Python representation of the network.

Each host, router, link, and flow object is denoted by a unique character id, and placed in a distinct dictionary. The check_id function checks the unique id constraint before construction any new objects. This is a global id constraint across all objects.

Parameters **bw** : *Blackwidow*

The simulation object containing settings and data recording.

Attributes

time	(float) The currenet simulation time.

Methods

add_event(event, delay)	Function to add an event to the queue
add_flow(flow_id, flow_src, flow_dest, ...)	Adds a flow to the network.
add_host(host_id)	Construct host and add to dictionary of hosts.
add_link(link_id, device_id1, device_id2, ...)	Adds a link to the network.
add_router(router_id)	Construct router and add to dictionary of routers.
check_id(obj_id)	Check if the id is not already used.
decrement_flows()	Decrements the number of active flows.
delete_device(device_id)	Deletes a device in the network.
delete_flow(flow_id)	Delete a flow from the network.
delete_link(link_id)	Deletes a link from the network.
dump([output])	Prints out network and returns networkx graph
empty()	Empties the event queue.
run()	Runs the network.
to_json()	Returns a JSON representation of the network.

add_event(*event*, *delay*)
Function to add an event to the queue

This function adds an event to the queue to be run after delay time.

Parameters **event** : *Event*

The event to be run.

delay : float

The amount of time in ms to wait before running the event.

add_flow(*flow_id*, *flow_src*, *flow_dest*, *data_amt*, *flow_start*)
Adds a flow to the network.

Parameters **flow_id** : string

A unique id for the flow.

flow_src : string

The id for the source *Device* for the flow.

flow_dest : string

The id for the destination *Device* for the flow.

data_amt : float

The amount of data for the flow to send in MB.

flow_start : float

The amount of time to wait before starting the flow in ms.

add_host(*host_id*)
Construct host and add to dictionary of hosts.

Parameters **host_id** : string

A unique id for the host.

add_link(*link_id*, *device_id1*, *device_id2*, *delay*, *rate*, *capacity*)
Adds a link to the network.

Parameters **link_id** : string

A unique id for the link.

device_id1 : string

The id of one of the *Device* objects to connect to the link.

device_id2 : string

The id of one of the *Device* objects to connect to the link.

delay : float

The propagation delay of the link in ms.

rate : float

The rate at which the link can send a packet in Mbps.

capacity : int

The capacity of the link buffer in KB.

add_router(*router_id*)

Construct router and add to dictionary of routers.

Parameters **router_id** : string

A unique id for the router.

check_id(*obj_id*)

Check if the id is not already used.

This function checks if the id is not already used. This function raises an exception if object id is not unique.

Parameters **obj_id** : string

The id to check.

decrement_flows()

Decrements the number of active flows.

delete_device(*device_id*)

Deletes a device in the network.

Parameters **device_id** : string

The id of the *Device* to delete.

delete_flow(*flow_id*)

Delete a flow from the network.

Parameters **flow_id** : string

The id of the flow to delete.

delete_link(*link_id*)

Deletes a link from the network.

Parameters **link_id** : string

The id of the link to delete.

dump(*output=False*)

Prints out network and returns networkx graph

Prints the devices, links, and flows associated with the network, and returns a pydot object with the network graph.

Parameters **output** : boolean, optional

Specifies whether to print the network information (the default is False).

Returns pydot

pydot object containing the network graph

empty()
Empties the event queue.

run()
Runs the network.

Dequeues events from the queue and runs them in order until the queue is empty or there are 0 flows active.

Returns **time** : int

The amount of time taken for the network to run.

to_json()
Returns a JSON representation of the network.

blackwidow.network.packet module

class blackwidow.network.packet.**AckPacket**(*packet_id*, *src*, *dest*, *flow_id*, *next_expected_id=0*, *timestamp=0*)

Bases: *blackwidow.network.packet.Packet*

Class for acknowledgement packets

Parameters **packet_id** : int

A unique id for the packet within a flow

src : Device

The device a packet originated from

dest : Device

The destination of the packet

flow_id : string

The flow_id of the flow this packet is in

next_expected_id : int

The next packet that the destination expects from the source.

timestamp : float, optional

The default value is 0 when this parameter is not used. This is used to track when the packet or the packet this is associated with if it is an ack was sent. This parameter is used to calculate round trip time in flow.

Attributes

pack_id	(int) The packet id or the id of the packet an ack is associated with.
src	(Device) The device a packet originated from
dest	(Device) The destination of the packet
flow_id	(string) The flow_id of the flow this packet is in
times-tamp	(float, optional) The default value is 0 when this parameter is not used. This is used to track when the packet or the packet this is associated with if it is an ack was sent. This parameter is used to calculate round trip time in flow.
is_ack	(boolean) True if ack packet; False otherwise.
is_routing	(boolean) True if routing packet; False otherwise.
size	(int) Size in bits of packet.
next_expected_id	(int) The next packet that the destination expects from the source.

class blackwidow.network.packet.**DataPacket**(*packet_id*, *src*, *dest*, *flow_id*, *timestamp=0*)

Bases: *blackwidow.network.packet.Packet*

Class for data packets

Parameters **packet_id** : int

A unique id for the packet within a flow

src : Device

The device a packet originated from

dest : Device

The destination of the packet

flow_id : string

The flow_id of the flow this packet is in

timestamp : float, optional

The default value is 0 when this parameter is not used. This is used to track when the packet or the packet this is associated with if it is an ack was sent. This parameter is used to calculate round trip time in flow.

Attributes

pack_id	(int) The packet id or the id of the packet an ack is associated with.
src	(Device) The device a packet originated from
dest	(Device) The destination of the packet
flow_id	(string) The flow_id of the flow this packet is in
times-tamp	(float, optional) The default value is 0 when this parameter is not used. This is used to track when the packet or the packet this is associated with if it is an ack was sent. This parameter is used to calculate round trip time in flow.
is_ack	(boolean) True if ack packet; False otherwise.
is_routing	(boolean) True if routing packet; False otherwise.
size	(int) Size in bits of packet.

class blackwidow.network.packet.**Packet**(*packet_id*, *src*, *dest*, *flow_id*, *timestamp=0*)

Bases: object

Super class for DataPackets and AckPackets

Parameters **packet_id** : int

A unique id for the packet within a flow

src : Device

The device a packet originated from

dest : Device

The destination of the packet

flow_id : string

The flow_id of the flow this packet is in

timestamp : float, optional

The default value is 0 when this parameter is not used. This is used to track when the packet or the packet this is associated with if it is an ack was sent. This parameter is used to calculate round trip time in flow.

Attributes

pack_id	(int) The packet id or the id of the packet an ack is associated with.
src	(Device) The device a packet originated from
dest	(Device) The destination of the packet
flow_id	(string) The flow_id of the flow this packet is in
times-tamp	(float, optional) The default value is 0 when this parameter is not used. This is used to track when the packet or the packet this is associated with if it is an ack was sent. This parameter is used to calculate round trip time in flow.
is_ack	(boolean) True if ack packet; False otherwise.
is_routing	(boolean) True if routing packet; False otherwise.
size	(int) Size in bits of packet.

class blackwidow.network.packet.**RoutingPacket**(*packet_id*, *src*, *dest*, *flow_id*, *routing_table*, *size*)

Bases: *blackwidow.network.packet.Packet*

Class for routing packets

Parameters **packet_id** : int

A unique id for the packet within a flow

src : Device

The device a packet originated from

dest : Device

The destination of the packet

flow_id : string

The flow_id of the flow this packet is in

routing_table : dictionary

Routing table to be updated

Attributes

pack_id	(int) The packet id or the id of the packet an ack is associated with.
src	(Device) The device a packet originated from
dest	(Device) The destination of the packet
flow_id	(string) The flow_id of the flow this packet is in
timestamp	(float, optional) The default value is 0 when this parameter is not used. This is used to track when the packet or the packet this is associated with if it is an ack was sent. This parameter is used to calculate round trip time in flow.
is_ack	(boolean) True if ack packet; False otherwise.
is_routing	(boolean) True if routing packet; False otherwise.
size	(int) Size in bits of packet.
routing_table	(dictionary) Routing table to be updated

blackwidow.network.rate_graph module

class blackwidow.network.rate_graph.**Data**(*time*, *size*)
Bases: object

Class to represent an amount of data and the time it was sent. It is used as a priority queue object with time as the priority.

Parameters **time** : float

Represents the network time this object was transferred.

size : int

The number of bits this object represents

Attributes

time	(float) Represents the network time this object was transferred.
size	(int) The number of bits this object represents

class blackwidow.network.rate_graph.**Rate_Graph**(*object_id*, *name*, *env*, *bw*)
Bases: object

Class to graph rates.

Parameters **object_id** : string

The id of the object recording.

name : string

The name of this Rate_Graph. Should specify which flow, link, or device is using it.

env : *Network*

The network that the flow belongs to.

bw : Blackwidow

The printer to print data to

Attributes

object_id	(string) The id of the object recording.
name	(string) The name of this Rate_Graph. Should specify which flow, link, or device is using it.
env	(*Network*) The network that the flow belongs to.
bw	(Blackwidow) The printer to print data to
window_size	(float) ms to average over
bits_in_window	(int) Number of bits that were sent in the last window_size ms
interval	(float) Record a data point every interval ms.
window	(PriorityQueue) Keeps track of each object recorded within last window_size ms.

Methods

add_point(packet, time)	Adds a point to the queue Parameters ———- packet : *Packet* The packet which was sent or received.
graph()	Graphs current rate
peek_time()	Return the time of the first object in the queue
remove_points(time)	Removes data before time Parameters ———- time : float The network's current time.

add_point (*packet*, *time*)
Adds a point to the queue Parameters ———- packet : *Packet*

The packet which was sent or received.

time [float] The network's current time.

graph ()
Graphs current rate

peek_time ()
Return the time of the first object in the queue

remove_points (*time*)
Removes data before time Parameters ———- time : float

The network's current time.

blackwidow.network.reno_flow module

class blackwidow.network.reno_flow.**RenoFlow** (*flow_id*, *source*, *destination*, *amount*, *env*, *time*, *bw*)
Bases: *blackwidow.network.tahoe_flow.TahoeFlow*

Implements TCP Reno. Adds Fast Retransmit and Fast Recovery Flows will trigger host behavior. Has slow start and congestion avoidance.

Parameters **flow_id** : string

A unique id for the flow.

source : *Device*

The source for the flow.

destination : *Device*

The destination for the flow.

amount : int

The amount of data to send in MB.

env : *Network*

The network that the flow belongs to.

time : float

The amount of time to wait before starting to send in ms.

bw : Blackwidow

The printer to print data to

Attributes

flow_id	(string) The flow id.
src	(*Device*) The source for the flow.
dest	(*Device*) The destination for the flow.
amount	(int) The amount of data left to send in MB.
env	(*Network*) The network that the flow belongs to.
flow_start	(float) The amount of time to wait before starting to send. Specified in ms.
pack_num	(int) The next pack_num to check to send.
cwnd	(float) Congestion window size.
ssthresh	(float) Slow start threshold
re-send_time	(float) ms before packets are sent after an ack receival
min_RTT	(float) Minimum round trip time observed for this flow
last_RTT	(float) Last round trip time observed for this flow
SRTT	(float) Weighted average of round trip times biased towards recent RTT
RTTVAR	(float) Variance of round trip times
RTO	(float) Retransmission timeout in ms
pack-ets_sent	(list) List of packets that have been sent but haven't had their ack received
pack-ets_time_out	(list) List of packets that have exceeded timeout and need to be resent
acks_arrived	(set) Set of ack packets that have been received
done	(int) 0 if flow isn't finished; 1 if flow is finished Used to avoid decrementing flow more than once.
send_rate	(Rate_Graph) Keeps track of the rate the flow is sending at and outputs to CSV file in real time.
re-ceive_rate	(Rate_Graph) Keeps track of the rate the flow is receiving at and outputs to CSV file in real time.
pack-ets_arrived	(list) Keeps track of packets(not acks) that have not arrived. Filled with all possible packet numbers when the flow starts and numbers are removed as each packet reaches the destination
to-tal_num_packets	(int) Total number of packets that need to be sent
last_pack_rec	(int) Packet number of previous next packet expected by the destination
counter	(int) Keeps track of duplicate acknowledgements

Methods

receive(packet)	Generate an ack or respond to bad packet.

_reset_window()
Called when a packet timeout occurs. Sets ssthresh to max(2, cwnd/2) and cwnd to 1. Resets counter

_send_ack(*packet*)
Creates ack for packet.

receive(*packet*)
Generate an ack or respond to bad packet. Parameters ———- packet : *Packet*

The packet to be received.

blackwidow.network.router module

class blackwidow.network.router.**Router**(*router_id*, *env*, *bw*)
Bases: *blackwidow.network.device.Device*

Class for routers.

Routers are responsible for initializing and updating their routing table, and sending packets based on their routing table.

Parameters **router_id** : string

A unique id for the router.

Attributes

network_id	(string) A unique id of the device in the network.
links	(list) A list of links that the router is connected to.
routing_table	(dict) A dictionary representing the router's routing table.
new_routing_table	(dict) A dictionary representing the router's new routing table.
env	(*Network*) The network that the link belongs to.
bw	(*Blackwidow*) BlackWidow simulation object containing simulation settings.
send_rate	(Rate_Graph object) Send rate graphing object.
receive_rate	(Rate_Graph object) Receive rate graphing object.

Methods

add_link(link)	Adds a link to the router.
send(packet)	Sends a packet to a link.
receive(packet)	Receives a packet from a link.
start_new_routing()	Starts a new routing round.
send_routing()	Sends a routing packet to all neighbors.
update_route()	Update the new_routing_table based on routing packets.
_distance(link)	Gets the distance of a link.

_distance(*link*)
Get the distance of the link.

Parameters **link** : Link

Link to get distance of.

add_link(*link*)
Overrides Device.add_link() to add to routing table.

Parameters **link** : Link

The link to add to the router.

receive(*packet*)
Process packet by sending it out.

If the packet is routing, calls update_route to update the new_routing_table.

Parameters **packet** : Packet

Received packet.

send(*packet*)
Send packet to appropriate link.

First looks in the new routing table to see if we know how to reach it there. Otherwise uses the old routing table.

Parameters **packet** : Packet

Packet to send through the router.

send_routing()
Send routing packets to all neighbors.

start_new_routing()
Start a new routing round.

If there is dynamic routing, updates the routing table to the new routing table built up by dynamic routing and measures the distance for each link.

update_route(*packet*)
Update routing table.

Goes through the routing table contained in the routing packet and determines if it contains a better way to get to each destination. This uses a distributed version of the Bellman-Ford algorithm.

Parameters **packet** : Packet

Routing packet to update the route.

blackwidow.network.tahoe_flow module

class blackwidow.network.tahoe_flow.**TahoeFlow**(*flow_id*, *source*, *destination*, *amount*, *env*, *time*, *bw*)
Bases: *blackwidow.network.flow.Flow*

Implements TCP Tahoe. Flows will trigger host behavior. Slow start and congestion avoidance already implemented in Flow. Just sets parameters for TCP Tahoe

Parameters **flow_id** : string

A unique id for the flow.

source : *Device*

The source for the flow.

destination : *Device*

The destination for the flow.

amount : int

The amount of data to send in MB.

env : *Network*

The network that the flow belongs to.

time : float

The amount of time to wait before starting to send in ms.

bw : Blackwidow

The printer to print data to

Attributes

flow_id	(string) The flow id.
src	(*Device*) The source for the flow.
dest	(*Device*) The destination for the flow.
amount	(int) The amount of data left to send in MB.
env	(*Network*) The network that the flow belongs to.
flow_start	(float) The amount of time to wait before starting to send. Specified in ms.
pack_num	(int) The next pack_num to check to send.
cwnd	(float) Congestion window size.
ssthresh	(float) Slow start threshold
resend_time	(float) ms before packets are sent after an ack receival
min_RTT	(float) Minimum round trip time observed for this flow
last_RTT	(float) Last round trip time observed for this flow
SRTT	(float) Weighted average of round trip times biased towards recent RTT
RTTVAR	(float) Variance of round trip times
RTO	(float) Retransmission timeout in ms
packets_sent	(list) List of packets that have been sent but haven't had their ack received
packets_time_out	(list) List of packets that have exceeded timeout and need to be resent
acks_arrived	(set) Set of ack packets that have been received
done	(int) 0 if flow isn't finished; 1 if flow is finished Used to avoid decrementing flow more than once.
send_rate	(Rate_Graph) Keeps track of the rate the flow is sending at and outputs to CSV file in real time.
receive_rate	(Rate_Graph) Keeps track of the rate the flow is receiving at and outputs to CSV file in real time.

Methods

1.1.3 blackwidow.parser package

Submodules

blackwidow.parser.parser module

blackwidow.parser.parser.**config_network**(*filename*, *bw*)
Returns Network object after parsing a .json file.

Parameters **filename** : string

name of .json file to configure off of.

bw : *BlackWidow*

BlackWidow simulation object containing simulation settings.

1.2 Submodules

1.3 blackwidow.blackwidow module

class blackwidow.blackwidow.**BlackWidow**(*settings={}*)
Bases: object

Runs simulation based on settings.

Generalizes Python's print to adapt to custom settings, and direct different types of messages to different outputs including files, or functions that dynamically generate graphs.

Parameters **settings** : dict

Contains settings to initialize the printer. Values include:

real_time [bool] Whether to graph in real time or write to files.

show_verbose [bool] Whether to print statements labelled verbose.

log_file [str] Name of file to write to. This is the prefix for all data types written to files. See documentation for write for more information.

data_dir [str] Directory where the data is stored.

static_routing [bool] Whether to use static routing.

routing_packet_size [int] Size of routing packets.

tcp_alg [str] Which TCP algorithm to use. Must be 'Reno', 'Fast', or 'Tahoe'.

Methods

run(file_name)	Runs the overall simulation based on settings specified when the BlackWidow object is constructed.
print_verbose(msg)	Handles a verbose message based on specified settings.
record(data, data_type)	Records data based on specified settings.
Examples	
`>>> from blackwidow import BlackWidow`	
`>>> settings = {'filename': 'case0.json' ... }`	
`>>> bw = BlackWidow(settings)`	
`>>> bw.run()`	

print_verbose(*msg*)
Handles a verbose message based on specified settings.

Parameters **msg** : str

Message to show.

record(*data*, *data_type*)
Records data based on specified settings.

Parameters **data** : str

Data point to record/plot.

data_type : str

Type of data, will be used as a file extension.

Notes

Standard data types: link.drop - "Time in ms", "Number of drops" link.sent - "Time in ms", "Number of packets sent" flow.window - "Time in ms", "Window size" flow.sent - "Time in ms", "Mega bits" flow.delay - "Time in ms", "Delay in ms"

run(*file_name*)
Runs the overall simulation based on settings specified when the BlackWidow object is constructed.

Parameters **file_name** : string

Name of config file containing network.

Returns **sim_time** : float

The amount of time taken for the network to finish running.

run_network(*network*)
Runs the overall simulation based on settings specified when the BlackWidow object is constructed.

Parameters **network** : *Network*

The network to run.

Returns **sim_time** : float

The amount of time taken for the network to finish running.

write()

Writes data to files.

This function writes each type of data to a file. The files are dependent on the extensions used to save data and the log_file file. Files are created as:

[log_file].[data_type].csv

Files are created in the data_dir directory in CSV format.

CHAPTER 2

run_interactive module

class run_interactive.**BlackWidowInteractive**(*completekey='tab'*, *stdin=None*, *stdout=None*)

Bases: cmd.Cmd

Command module to run the simulator in interactive mode.

This class runs the simulator in interactive mode and supports various command.

Attributes

intro	

Methods

create_network([settings, f])	Initializes the network and bw variables.
default(line)	Overrides the default method on the base class to provide shortcut aliases for comman
do_EOF(line)	Ends the program.
do_add_flow(line)	Adds a flow.
do_add_host(line)	Adds multiple hosts.
do_add_link(line)	Adds a link.
do_add_router(line)	Adds multiple routers.
do_clear(line)	Clears the graph.
do_close(line)	Closes the graph.
do_delete_device(line)	Deletes multiple devices.
do_delete_flow(line)	Deletes mulitple flows.
do_delete_link(line)	Deletes multiple links.
do_dump(line)	Saves the network to a file.
do_exit(line)	Ends the program.
do_load(line)	Loads a file.
do_reset(line)	Resets network
do_reset_v(line)	Resets parameters for interactive
do_run(line)	Runs the network.
do_set_dpi(line)	Sets the dpi to show the network.
do_set_output(line)	Sets network graph textual behavior.
do_set_proj(line)	Sets the projection to show the network.
do_set_routing_packet_size(line)	Sets routing packet size.
do_set_show(line)	Sets network graph display behavior.

Continued on next pag

Table 2.1 – continued from previous page

do_set_static_routing(line)	Sets static routing.
do_set_tcp_alg(line)	Sets TCP algorithm.
do_set_verbose(line)	Sets verbose output.
do_show(line)	Shows the network.
do_stop(line)	Stops the network.
help_EOF()	Prints help message for EOF command
help_add_flow()	Prints help message for add_flow command
help_add_host()	Prints help message for add_host command
help_add_link()	Prints help message for add_link command
help_add_router()	Prints help message for add_router command
help_clear()	Prints help message for clear command
help_close()	Prints help message for close command
help_delete_device()	Prints help message for delete_device command
help_delete_flow()	Prints help message for delete_flow command
help_delete_link()	Prints help message for delete_link command
help_dump()	Prints help message for dump command
help_exit()	Prints help message for exit command
help_load()	Prints help message for load command
help_reset()	Prints help message for reset command
help_reset_v()	Prints help message for reset_v command
help_run()	Prints help message for run command
help_set_dpi()	Prints help message for set_dpi command
help_set_output()	Prints help message for set_output command
help_set_proj()	Prints help message for set_proj command
help_set_routing_packet_size()	Prints help message for set_routing_packet_size command
help_set_show()	Prints help message for set_show command
help_set_static_routing()	Prints help message for set_static_routing command
help_set_tcp_alg()	Prints help message for set_tcp_alg command
help_set_verbose()	Prints help message for set_verbose command
help_show()	Prints help message for show command
help_stop()	Prints help message for stop command

create_network(*settings=None*, *f=None*)

Initializes the network and bw variables.

Parameters **settings** : dict, optional

A dictionary of settings (the default is None). See *Blackwidow* for valid values.

f : string, optional

The filename containing the network (the default is None).

default(*line*)

Overrides the default method on the base class to provide shortcut aliases for commands.

Commands can be entered by typing partial commands that identify a command uniquely.

Parameters **line** : string

String containing command and argument

do_EOF(*line*)

Ends the program.

Parameters **line** : string

A string containing command line arguments. Ignored.

do_add_flow(*line*)
Adds a flow.

Parameters **line** : string

A string containing command line arguments. See help_add_flow.

do_add_host(*line*)
Adds multiple hosts.

Parameters **line** : string

A string containing command line arguments. See help_add_host.

do_add_link(*line*)
Adds a link.

Parameters **line** : string

A string containing command line arguments. See help_add_link.

do_add_router(*line*)
Adds multiple routers.

Parameters **line** : string

A string containing command line arguments. See help_add_router.

do_clear(*line*)
Clears the graph.

Parameters **line** : string

A string containing command line arguments. Ignored.

do_close(*line*)
Closes the graph.

Parameters **line** : string

A string containing command line arguments. Ignored.

do_delete_device(*line*)
Deletes multiple devices.

Parameters **line** : string

A string containing command line arguments. See help_delete_device.

do_delete_flow(*line*)
Deletes mulitple flows.

Parameters **line** : string

A string containing command line arguments. See help_delete_flow.

do_delete_link(*line*)
Deletes multiple links.

Parameters **line** : string

A string containing command line arguments. See help_delete_link.

do_dump(*line*)
Saves the network to a file.

Parameters **line** : string

A string containing command line arguments. See help_dump.

do_exit(*line*)
Ends the program.

Parameters **line** : string

A string containing command line arguments. Ignored.

do_load(*line*)
Loads a file.

Parameters **line** : string

A string containing command line arguments. See help_load.

do_reset(*line*)
Resets network

Parameters **line** : string

A string containing command line arguments. Ignored.

do_reset_v(*line*)
Resets parameters for interactive

Parameters **line** : string

A string containing command line arguments. Ignored.

do_run(*line*)
Runs the network.

Parameters **line** : string

A string containing command line arguments. Ignored.

do_set_dpi(*line*)
Sets the dpi to show the network.

Parameters **line** : string

A string containing command line arguments. See help_set_dpi.

do_set_output(*line*)
Sets network graph textual behavior.

Parameters **line** : string

A string containing command line arguments. See help_set_output.

do_set_proj(*line*)
Sets the projection to show the network.

Parameters **line** : string

A string containing command line arguments. See help_set_proj.

do_set_routing_packet_size(*line*)
Sets routing packet size.

Parameters **line** : string

A string containing command line arguments. See help_set_routing_packet_size.

do_set_show(*line*)
Sets network graph display behavior.

Parameters **line** : string

A string containing command line arguments. See help_set_show.

do_set_static_routing(*line*)
Sets static routing.

Parameters **line** : string

A string containing command line arguments. See help_set_static_routing.

do_set_tcp_alg(*line*)
Sets TCP algorithm.

Parameters **line** : string

A string containing command line arguments. See help_set_tcp_alg.

do_set_verbose(*line*)
Sets verbose output.

Parameters **line** : string

A string containing command line arguments. See help_set_verbose.

do_show(*line*)
Shows the network.

Parameters **line** : string

A string containing command line arguments. Ignored.

do_stop(*line*)
Stops the network.

Parameters **line** : string

A string containing command line arguments. Ignored.

help_EOF()
Prints help message for EOF command

help_add_flow()
Prints help message for add_flow command

help_add_host()
Prints help message for add_host command

help_add_link()
Prints help message for add_link command

help_add_router()
Prints help message for add_router command

help_clear()
Prints help message for clear command

help_close()
Prints help message for close command

help_delete_device()
Prints help message for delete_device command

help_delete_flow()
Prints help message for delete_flow command

help_delete_link()
Prints help message for delete_link command

help_dump()
Prints help message for dump command

help_exit()
Prints help message for exit command

help_load()
Prints help message for load command

help_reset()
Prints help message for reset command

help_reset_v()
Prints help message for reset_v command

help_run()
Prints help message for run command

help_set_dpi()
Prints help message for set_dpi command

help_set_output()
Prints help message for set_output command

help_set_proj()
Prints help message for set_proj command

help_set_routing_packet_size()
Prints help message for set_routing_packet_size command

help_set_show()
Prints help message for set_show command

help_set_static_routing()
Prints help message for set_static_routing command

help_set_tcp_alg()
Prints help message for set_tcp_alg command

help_set_verbose()
Prints help message for set_verbose command

help_show()
Prints help message for show command

help_stop()
Prints help message for stop command

run_interactive.**check_args**(*args*, *n*)
Checks the provided list of args.

Checks if the provided list of args has the correct number of args.

Parameters **args** : list

A list of strings.

n : int

The number of arguments that should be provided.

Returns boolean

Returns True if the number of args is correct, or False otherwise.

run_interactive.**create_bw**(*settings=None*, *f=None*)

Creates a command module and runs it.

Parameters **settings** : dict, optional

A dictionary of settings (the default is None).

f : string, optional

The filename containing the network (the default is None).

run_interactive.**main**()

CHAPTER 3

run_simulator module

Runs blackwidow simulator on a specified set of files.

This script parses user arguments and configures the blackwidow module to run based on user arguments.

run_simulator.**main**()
: Runs the simulator.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

b

r

Symbols

A

B

C

L

M

N

P

R

S

T

U

W