NEAT-TCP

Practical Evaluation of NEAT-TCP on a Wireless Testbed

TECHNISCHE UNIVERSITÄT DARMSTADT

SECURE MOBILE NETWORKING

10.08.18 | Kay Luis Wallaschek | kayluis.wallaschek@stud.tu-darmstadt.de

Outline

- Contributions
- Background
- Port of NEAT-TCP to Linux
- Building the Testbed
- Testbed Locations
- Evaluation
- Conclusion

Contributions

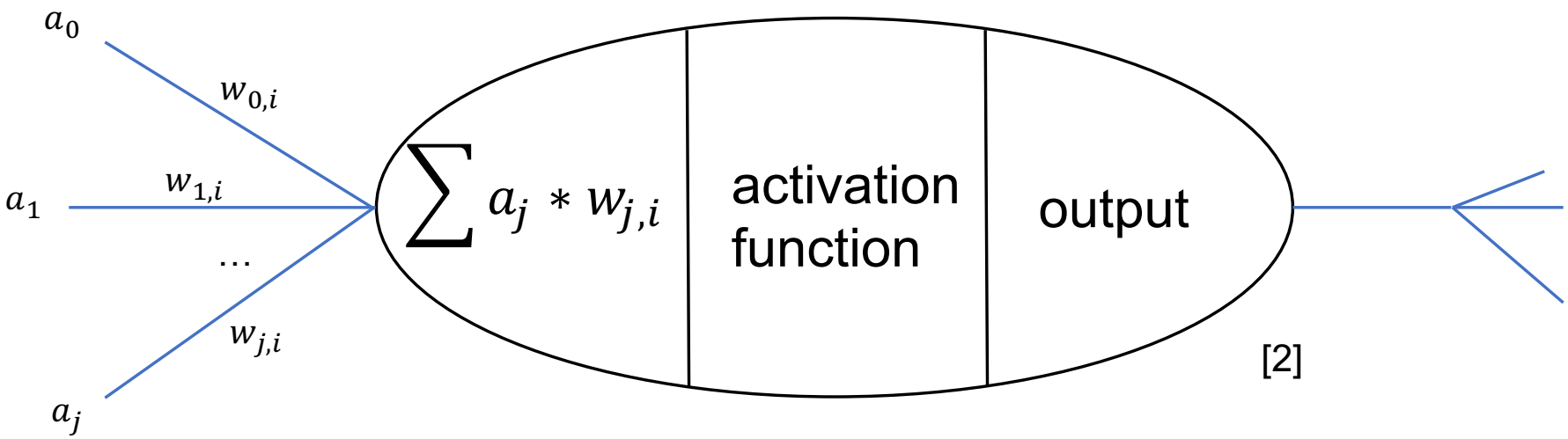
- Porting of NEAT-TCP from ns-3 to Linux as a pluggable kernel module.

- Planning and building of a testbed.

- Evaluation of NEAT-TCP and other Congestion Control algorithms on this testbed in comparison to ns-3 results.

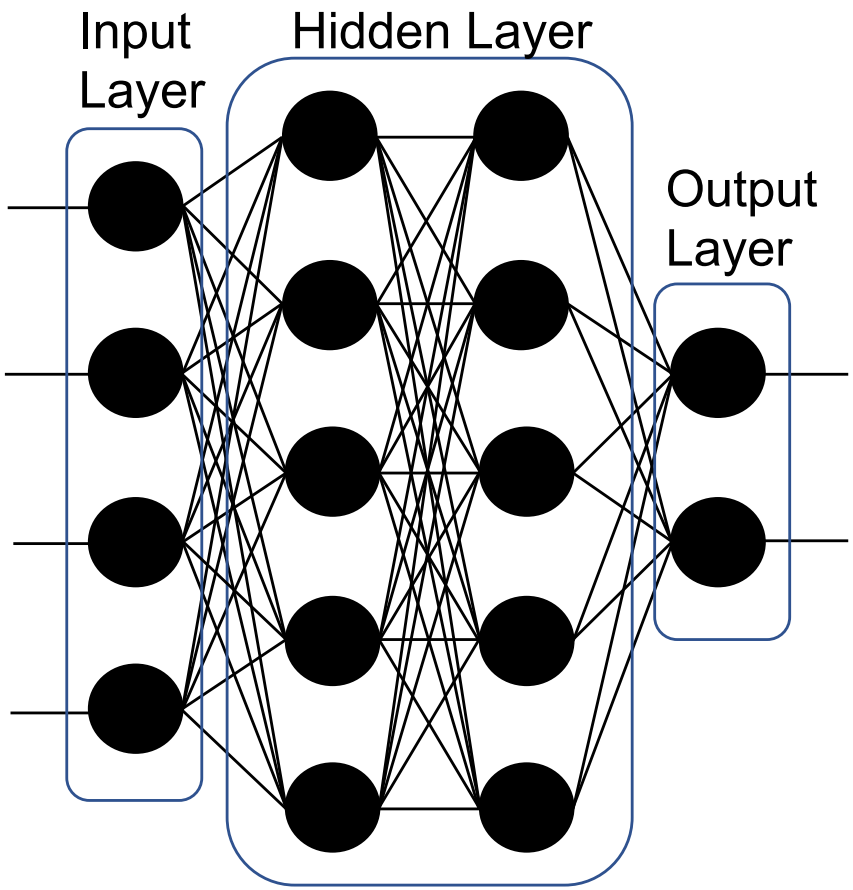
Neural Networks (1)

- McCulloch-Pitts Neuron Model:



Neural Networks (2)

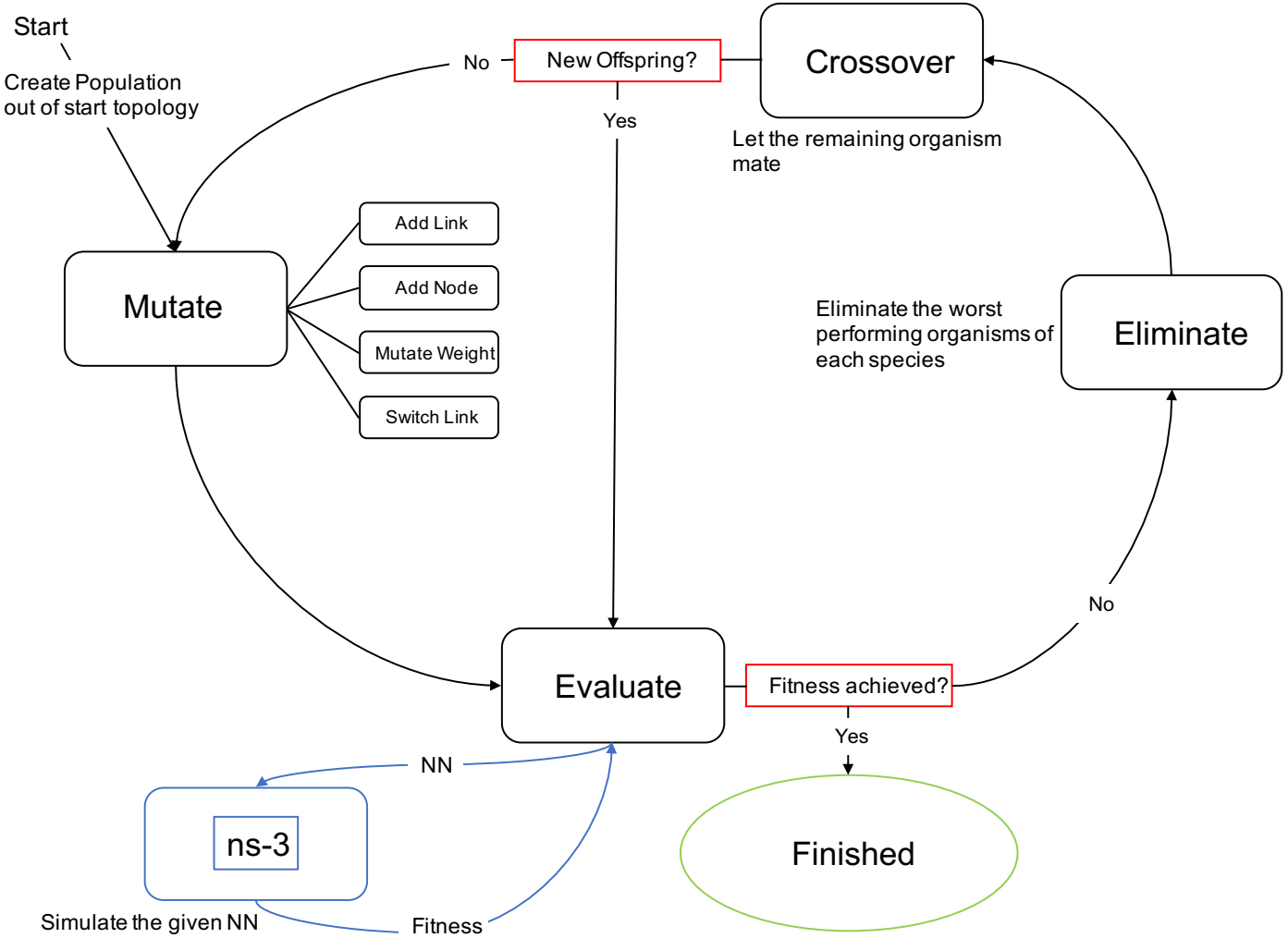
- Feed-Forward Neural Network:



NEAT

- NeuroEvolution of Augmenting Topologies
- Kenneth O. Stanley and Risto Miikkulainen (2002)
- Genetic algorithm (evolution)
- Prominent in machine learning for games
- For example: MarI/O
  - NEAT evolves NNs that play and finish Super Mario levels.

NEAT-TCP


Start
Create Population
out of start topology
Mutate
Add Link
Add Node
Mutate Weight
Switch Link
Evaluate
Fitness achieved?
Yes
No
Eliminate
Eliminate the worst
performing organisms of
each species
Crossover
Let the remaining organism
mate
New Offspring?
No
Yes
Finished
NN
ns-3
Simulate the given NN
Fitness

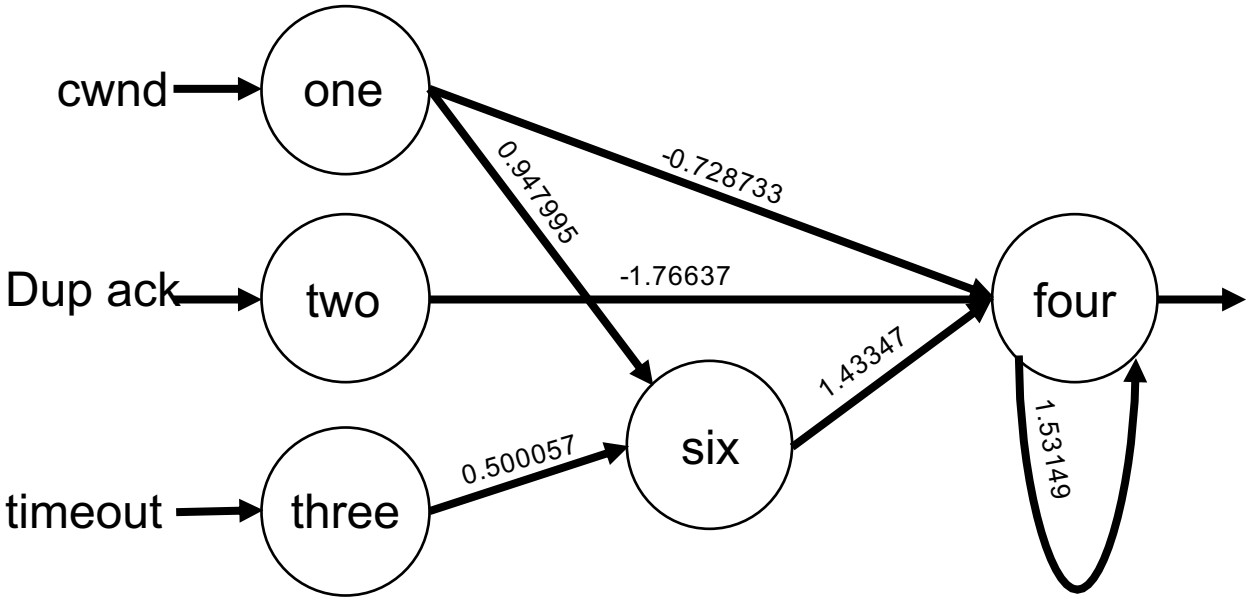
Pluggable Kernel Module

- Mandatory functions:
  - .ssthresh
  - .cong_avoid

- Additional functions:
  - .init
  - .cwnd_event
  - .undo_cwnd
  - .in_ack_event
  - tcp_neat_eval

Problems – Kernel Module

Pluggable congestion control kernel modules do not have access to FPUs.



Solution

Translate neurons to logical functions.
Possible because the activation function is binary.

Example:

$$\sigma(x) = \begin{cases} 1 & x > 0 \\ 0 & \text{else} \end{cases} \quad \text{with} \quad x = \vec{w} * \vec{x} = \sum_{i=0}^{n} w_i * x_i$$

For node six: $x_6 = 0.500057 * \sigma(x_{3,\text{in}}) + 0.947995 * \sigma(x_{1,\text{in}})$

$$x_6 = \sigma(x_{6,\text{in}}) = x_1 \vee x_3$$

Testbed (1)

SECURE MOBILE NETWORKING

- For each Raspberry Pi:
  - Install Raspbian
  - Update & Upgrade
  - Install iperf3
  - Enable IP-Forwarding:
    - In /etc/sysctl.conf -> net.ipv4.ip_forward = 1
  - Copy compiled congestion control kernel modules
  - Copy individual scripts
  - Configure crontab

Testbed (2)

- RPi interface configuration:

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

iface eth0 inet dhcp

auto wlan0
iface wlan0 inet static
  address 10.2.1.n
  netmask 255.255.255.0
  wireless-channel 1
  wireless-essid PiAdHocNetwork
  wireless-mode ad-hoc
  wireless-txpower p
up route add -host 10.2.1.k gw 10.2.1.l dev wlan0
```

Testbed (3)

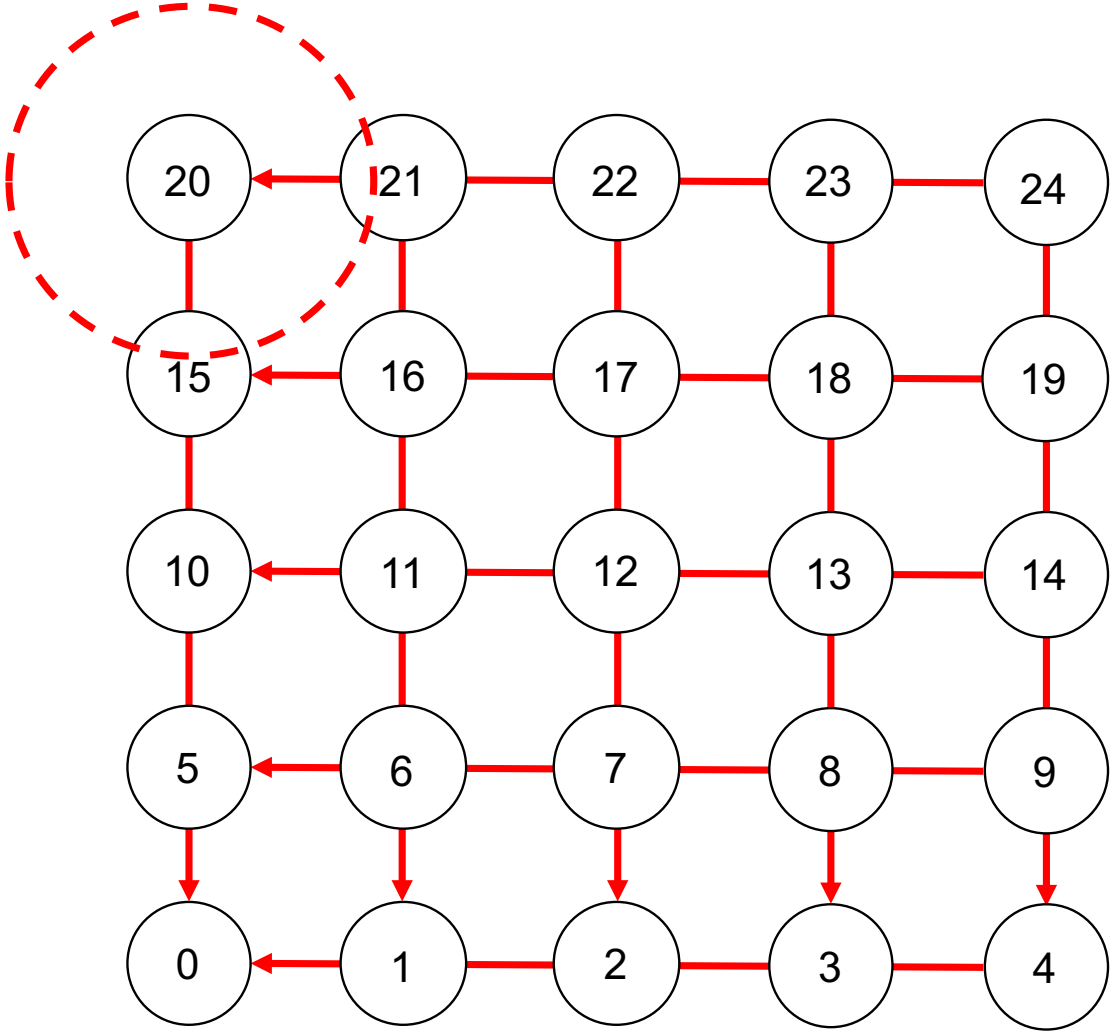
- Crontab to time scripts
  - @reboot
  - 15 12 * * *

- Experiment script:

```
#!/bin/bash
$ sudo sysctl -w net.ipv4.tcp_congestion_control=hybla
$ iperf3 -c 10.2.1.4 -V -i 1 -t 300 --logfile /home/pi/Desktop/
    output/hybla_23-to-3.log
```

Testbed (4)

Evaluation - Experimental Setup



Evaluation – Testsites (1)

Evaluation – Testsites (2)

- Free grassland
- No interferences
- Problems:
  - On that day 30° C and no clouds
  - Overheating RPis
    - Resulting in degraded performance
      - Lower transmission range

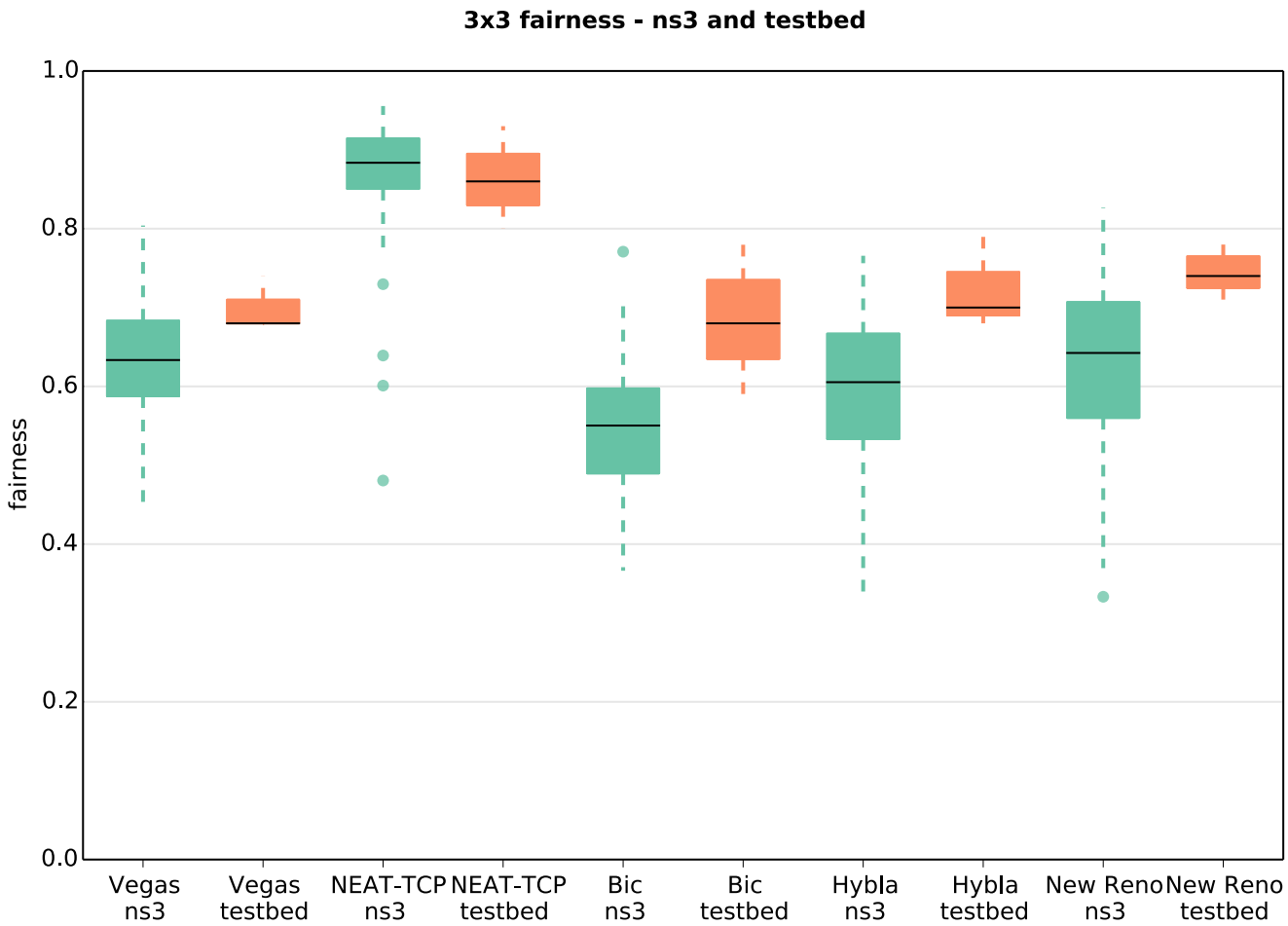
Evaluation – Testsites (3)

- L402/1
- Weird transmission characteristics
- No representative results

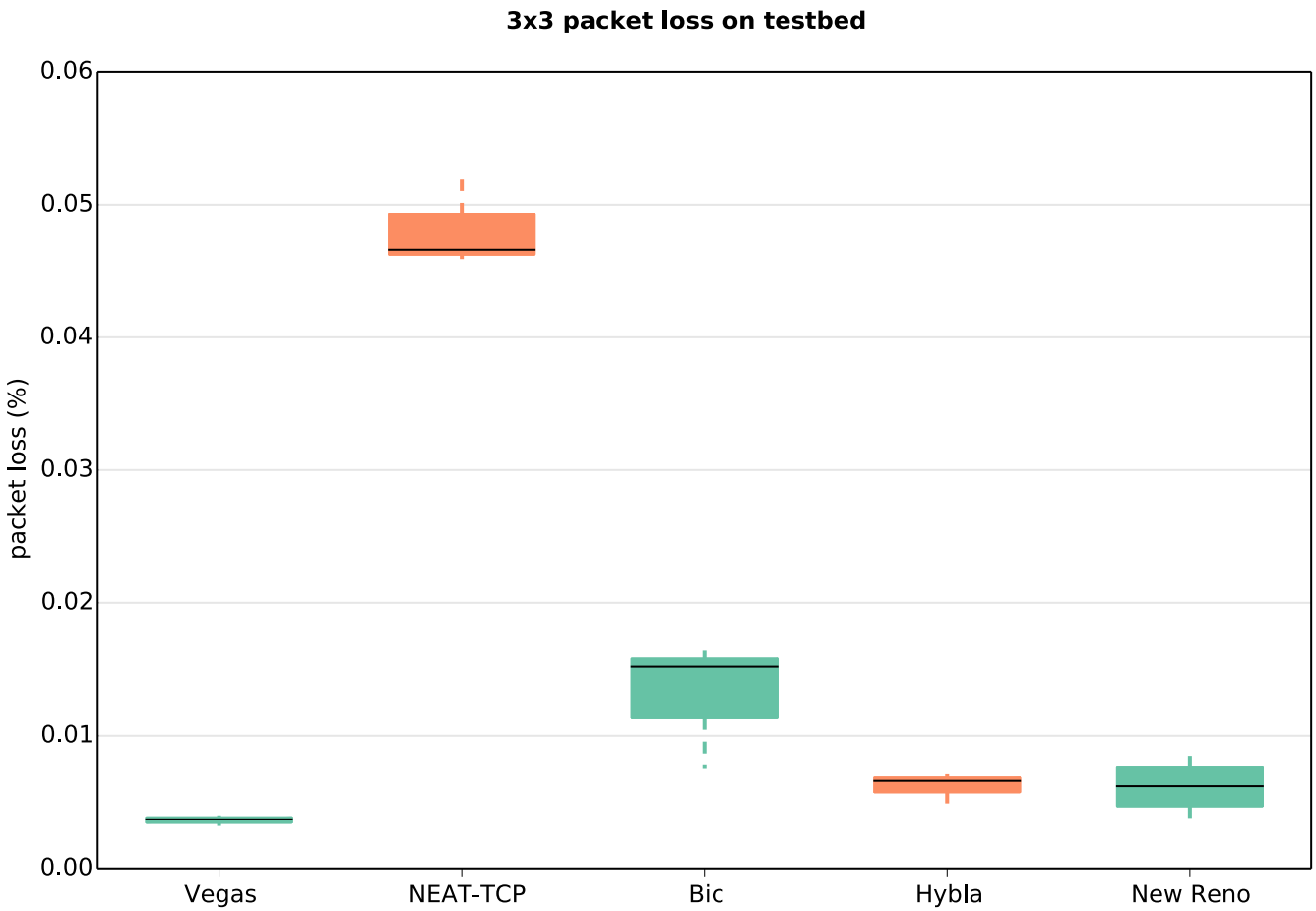
Evaluation – Testsites (4)

- S202/110
- Only 3x3
- Comparable Results
- Wireless-power reduced to 1dBm

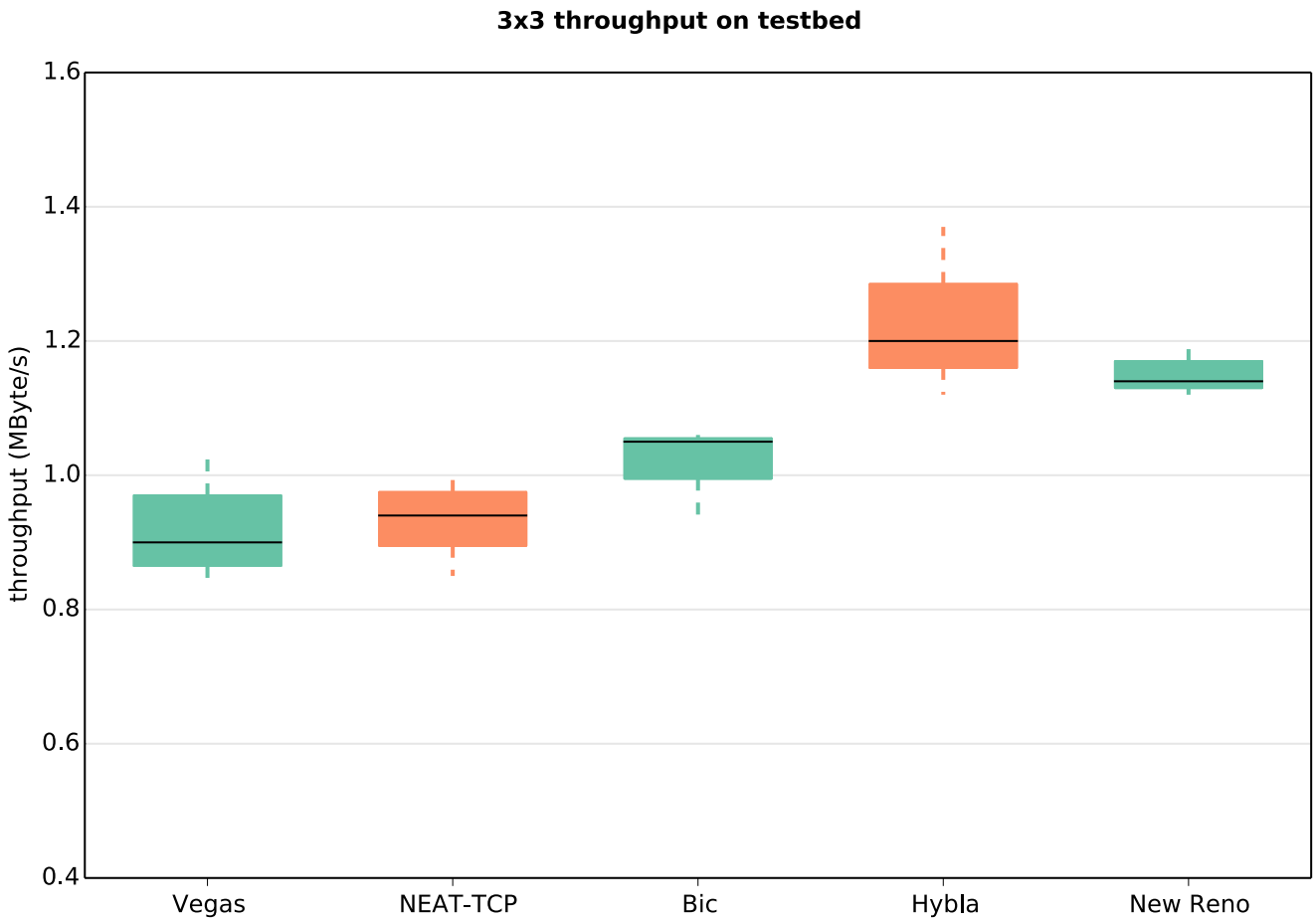
Evaluation on 3x3



Evaluation on 3x3



Evaluation on 3x3



Conclusion

SECURE MOBILE NETWORKING

- Built a working testbed of Raspberry Pis
- Ported NEAT-TCP to Linux as a pluggable kernel module
- Tested several configurations and locations
- Gathered meaningful results for the 3x3 configuration

Thank you for your attention!

[1] Sumit Rangwala, Apoorva Jindal, Ki-Young Jang, Konstantinos Psounis, and Ramesh Govindan. “"Understanding congestion control in multi-hop wireless mesh networks".” In: *MobiCom 08, Proceedings of the 14th ACM international conference on Mobile computing and networking* (2008), pp. 291–302.

[2] Warren McCulloch and William Pitts. “"A logical calculus of the ideas immanent in nervous activity".” In: *Bulletin of Mathematical Biophysics* (1943), pp. 115–133.

[4] Kenneth O. Stanley and Risto Miikkulainen. “Evolving Neural Networks Through Augmenting Topologies.” In: *Evolutionary Computation* 10.2 (2002), pp. 99–127. url: http://nn.cs.utexas. edu/?stanley:ec02.

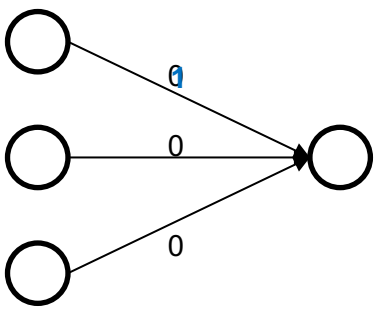
Copyright Notice

- This document has been distributed by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, notwithstanding that they have offered their works here electronically.

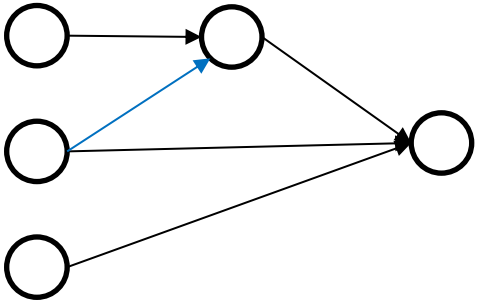
- It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

NEAT – Mutations

- Four different mutations:
  - Link Weight Mutation

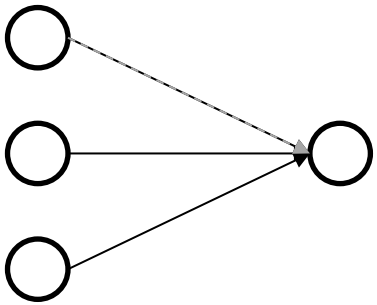


  - Add Link Mutation

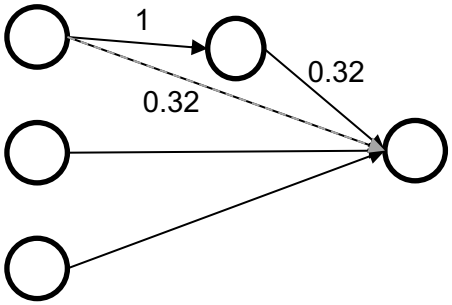


NEAT – Mutations

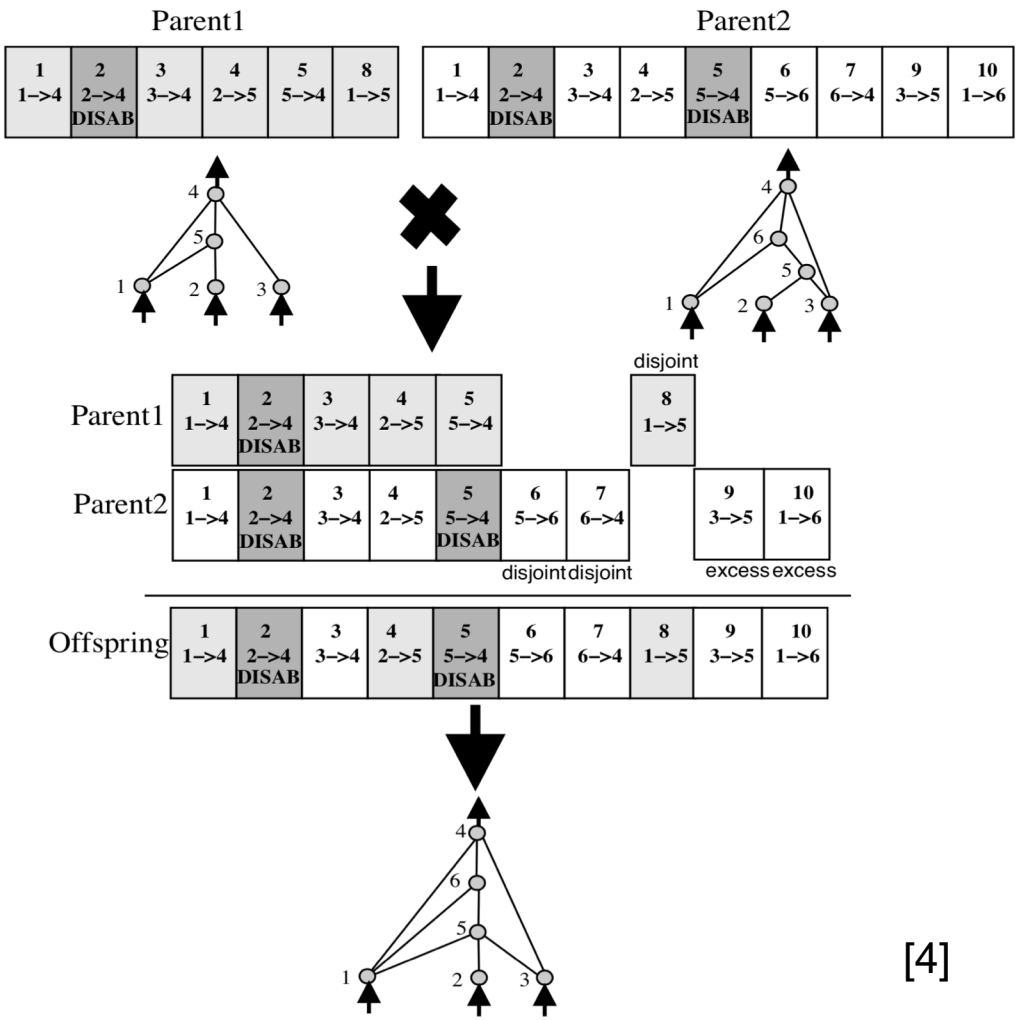
- Four different mutations:
  - Switch Link Mutation



  - Add Node Mutation

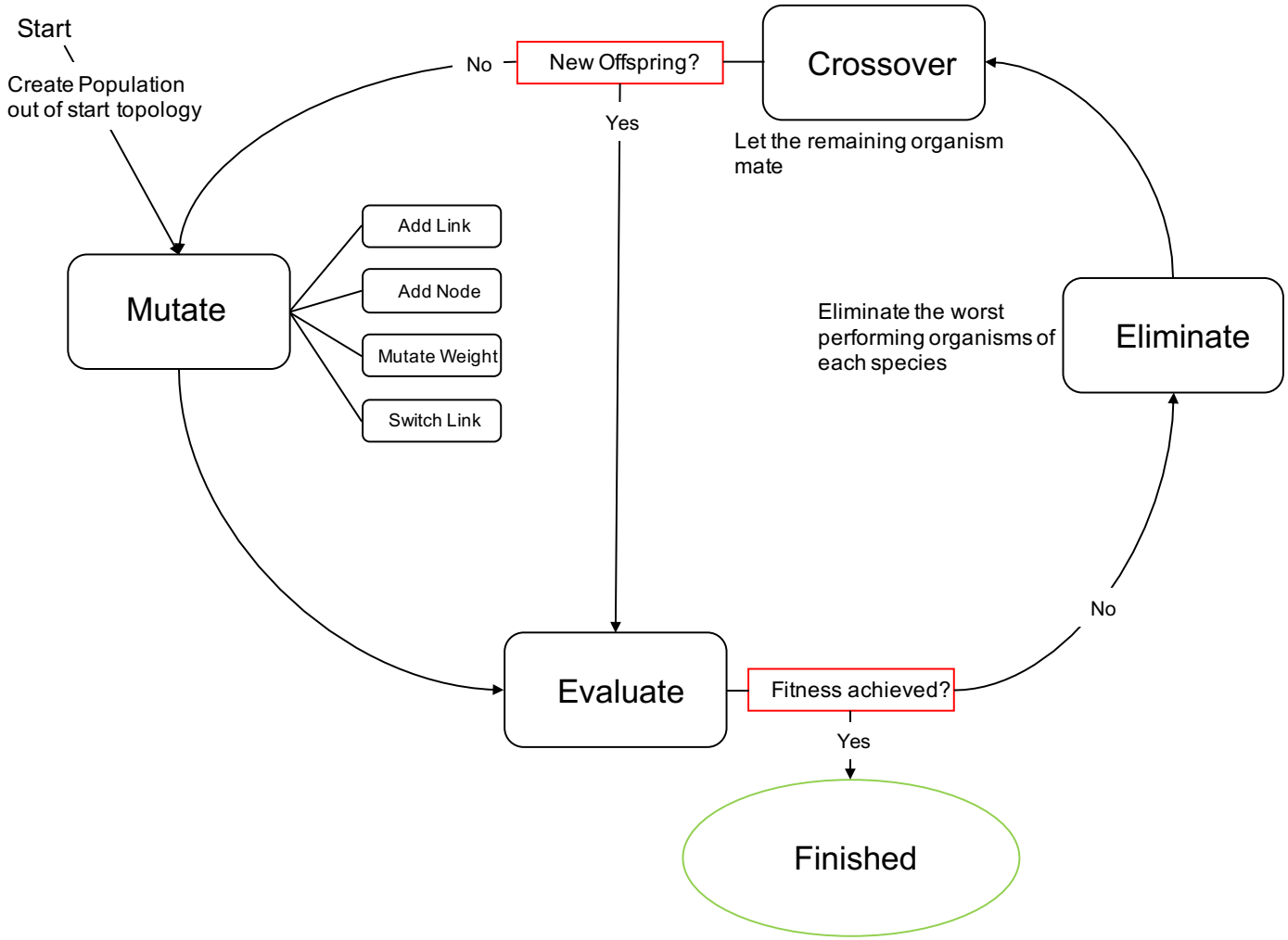


NEAT – Mating/Speciation



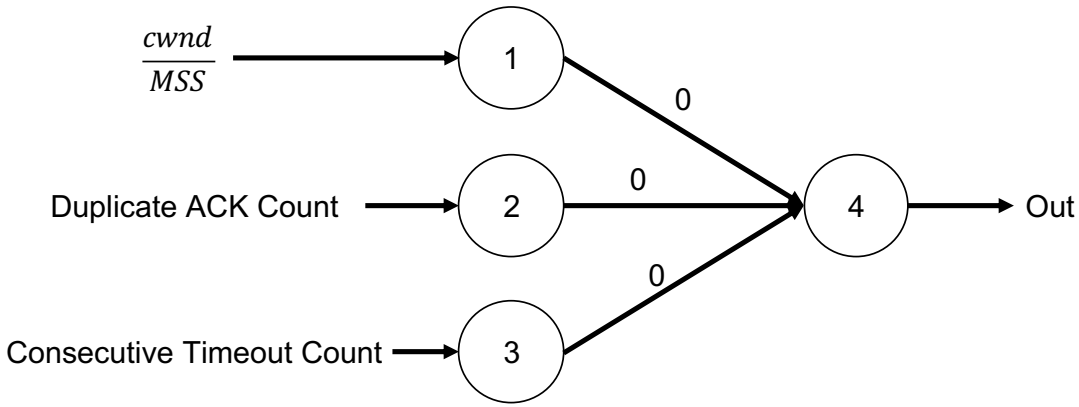
[4]

NEAT – Iteration



NEAT-TCP (2)

Initial NN:



Parameters:

- Recurrent Probability: 5%
- Mutate Link Probability: 80%
- Toggle Link Probability: 10%
- Add Node Probability: 1%
- Add Link Probability: 30%
- Population Size: 500 Organisms
- Dropoff Age: 15
- Epochs: 100

NEAT-TCP (3)

Activation Functions: $\phi_1(x) = \begin{cases} 1 & , x > 0 \\ 0 & , \text{else} \end{cases}$

$$\phi_2(x) = x$$

cwnd1: $\text{cwnd} = \text{cwnd} + \text{out} * \text{MSS}$

cwnd2: $\text{cwnd} = \text{out} * \text{MSS}$

cwnd3: $\text{cwnd} = \begin{cases} \text{cwnd} + \text{MSS}, & \text{out} > 0 \\ \text{cwnd} - \text{MSS}, & \text{else} \end{cases}$

NEAT-TCP (4)

Fitness Functions:

$$\rho_1 = \frac{\tau_a^\sigma}{\tau_t^\sigma} * \mathcal{J}$$

$$\mathcal{J}(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} {x_i}^2}$$

Jain‘s fairness index

$$\rho_2 = \tau_a^\sigma$$

Where:

- τ_a^σ = achieved total throughput
- τ_t^σ = target total throughput

NEAT-TCP (5)

- Time Points of Evaluation
  - PktAcked:
    - Every time a packet is acknowledged, the NN is evaluated and the cwnd is changed accordingly
  - Increase Window:
    - Every time New Reno would increase the cwnd, the NN is evaluated and the cwnd is changed accordingly

NEAT-TCP Convergence

- Three stop criteria:
  - Target fitness achieved
  - Number of epochs reached
  - Abortion by user

- Based on randomness

- Generally, with larger population:
  - Faster (in terms of epochs)
  - More structural minimal solutions
  - Higher chance of finding a solution

NEAT-TCP Applicability

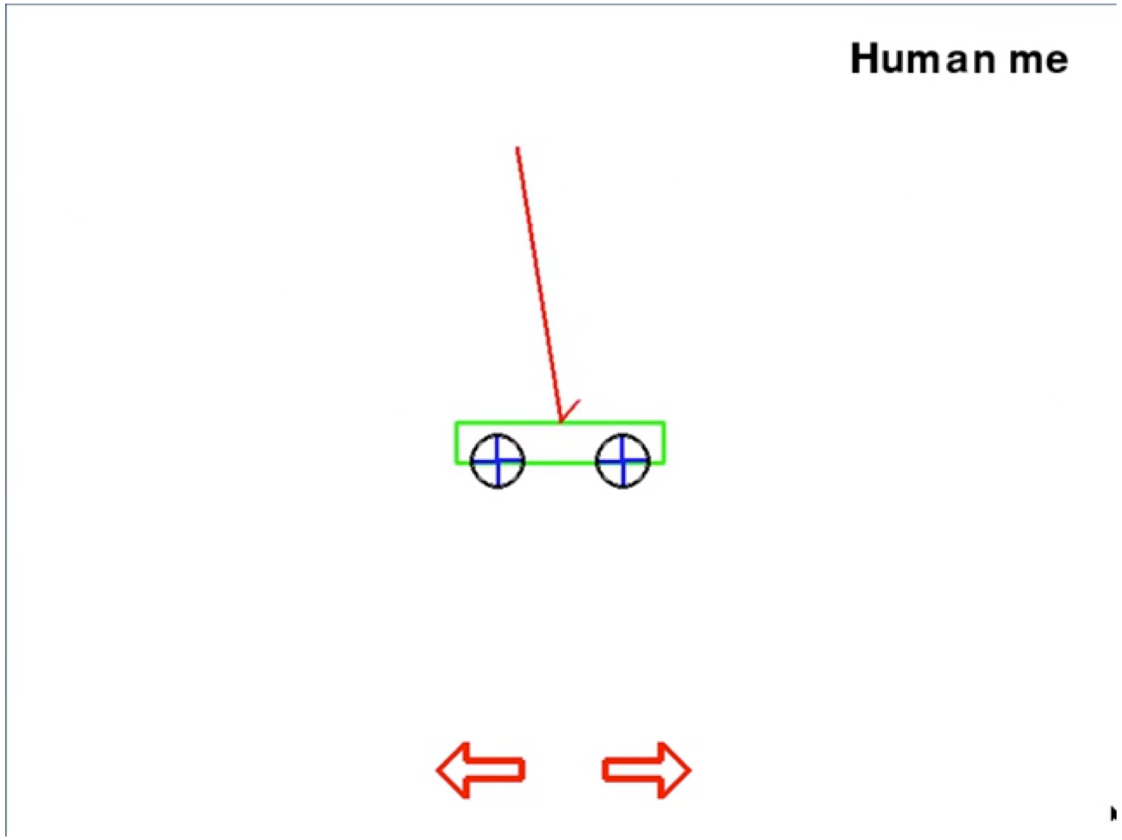
- Could work in other environments
  - Wired
  - Mobile Networks
  - Etc.

- NEAT-TCP works for a range of networks
  - (generated for 5x5 but has respectable performance in 3x3)
  - But not tested in different flow configurations

- Industrial/Commercial context
  - Generate a NEAT-TCP before shipping/installation of network

Why these parameters?

Same parameter file used as in double pole balancing on a cart:



[3]

Why is Congestion Control a difficult task in multi-hop WMNs?

- CC is triggered because of wrong assumption
- Congestion depends not only on a single node, but is a neighborhood phenomenon. (2 hop range)

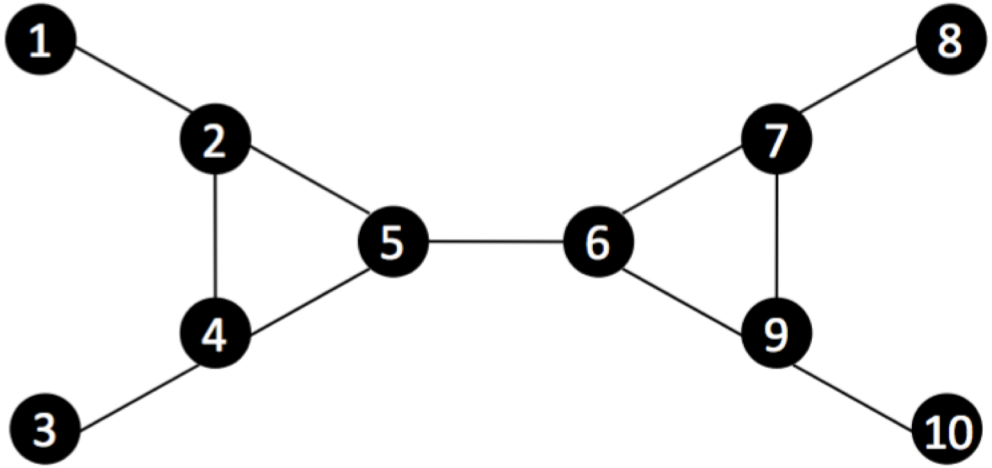


Figure 3: Congestion neighborhood

[1]

Why is Congestion Control a difficult task in multi-hop WMNs?

- Congestion depends not only on a single node, but is a neighborhood phenomenon. (2 hop range)

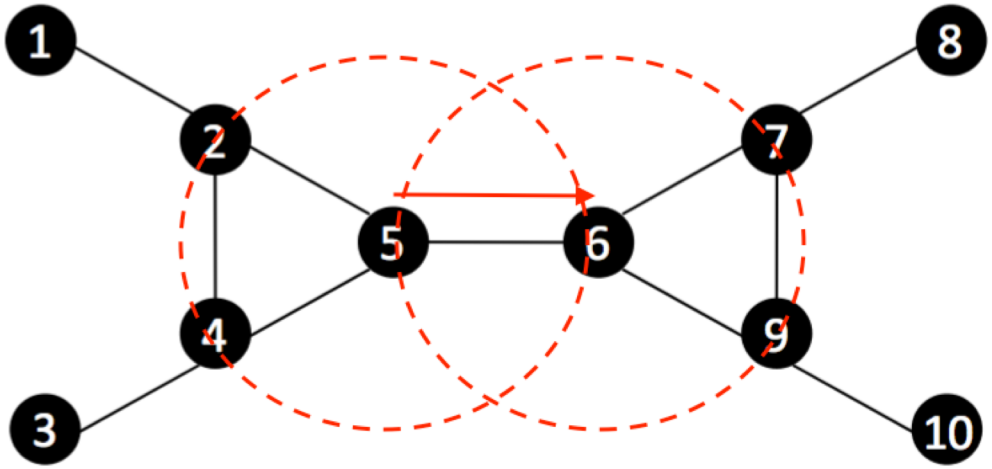


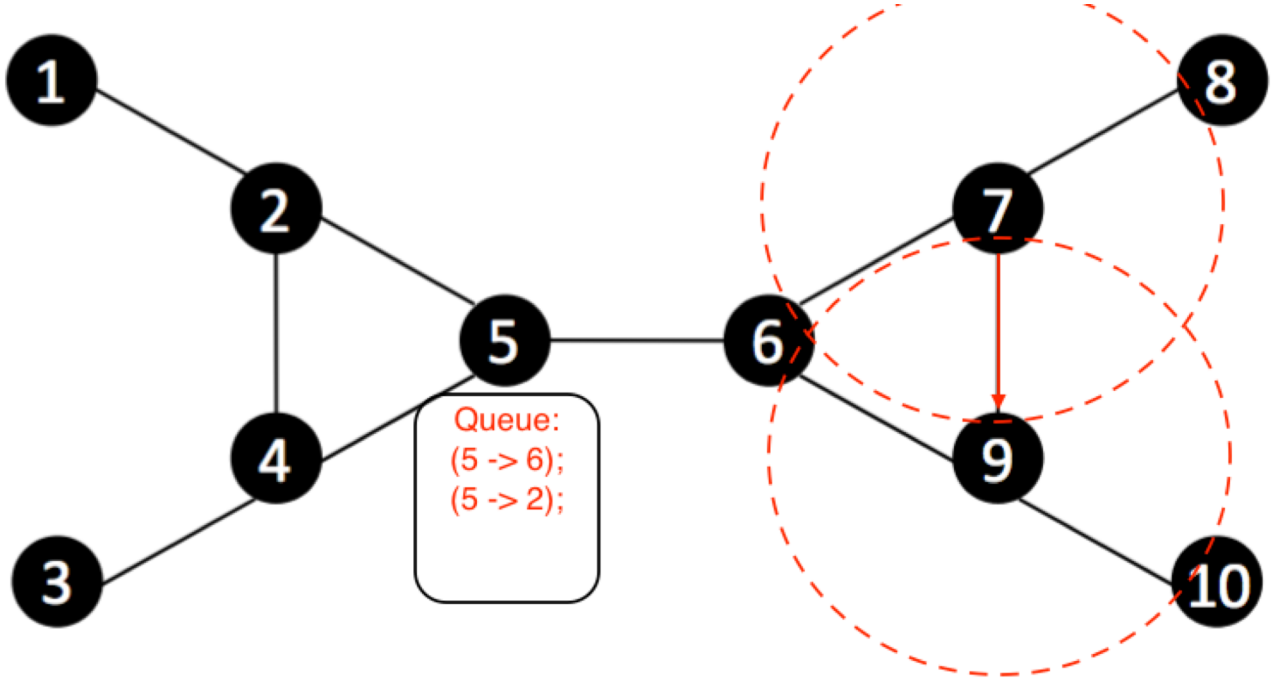
Figure 3: Congestion neighborhood

[1]

Challenges of TCP Congestion Control in Wireless Mesh networks

- Congestion is a neighborhood phenomenon.

- RTS/CTS -> Head of line blocking effect.



[1]