Water flowing model for load-balancing in the environment of Active Networks

- Active Networks
- Environment
- Virtual Topology Discovery
- Model
- Distribution Algorithm
- Expected Behavior
- Comparison
- Future Work

Active Networks

- Programmable networks built of NodeOS
- NodeOS is OS providing services independent of a specific active networking implementation
- NodeOS provides services to Execution Environment (EE) running on top of it
- EE provides AN protocol specific processing
- Packets carry executable code or reference to
- End system uses the packets to modify AN behavior to its favor and to create/refine protocols

Environment

- Active Networks supporting anycast
- Each EE supports storing of information with TTL
- Each EE provides list of its neighborhoods
- Each EE can decide whether removal of process will boost performance of other processes
- For dat processing while in transit a route can be discovered in advance //tracert
- End system has rights to dynamically upload and run code
- Each process has its own GUID

Virtual Topology Discovery

- Computation time is discrete one; time points T_i
- Task last T, T is ordered set of all T_i; T= $[T_i]$
- Su represents state of utilized EE
- A change of S_U invokes time transition

$$S_U \rightarrow S_U^{|} \neq S_U \Rightarrow T_i \rightarrow T_{i+1}$$

- Let α be total count of processes of application
- Let β be count of processes initiating communication at T_i
- Let γ be count of processes communicated to by one

Virtual Topology Discovery

- Mark a single process as *p* and the set of processes involved in communication on initiative of one process as

  $$\Gamma = [p]^{\gamma+1}$$

- Mark set of all processes communicating in one time point as Π

  $$\Pi = \bigcap_{\beta} \Gamma \quad \text{where} \quad \pi = |\Pi| \quad \text{and} \quad 0 \le \pi \le \alpha|_{Ti}$$

- Create directed graph with π nodes - 1 node 1 process of Π; direction represents initiator
- Merge these graphs to get used virtual topology

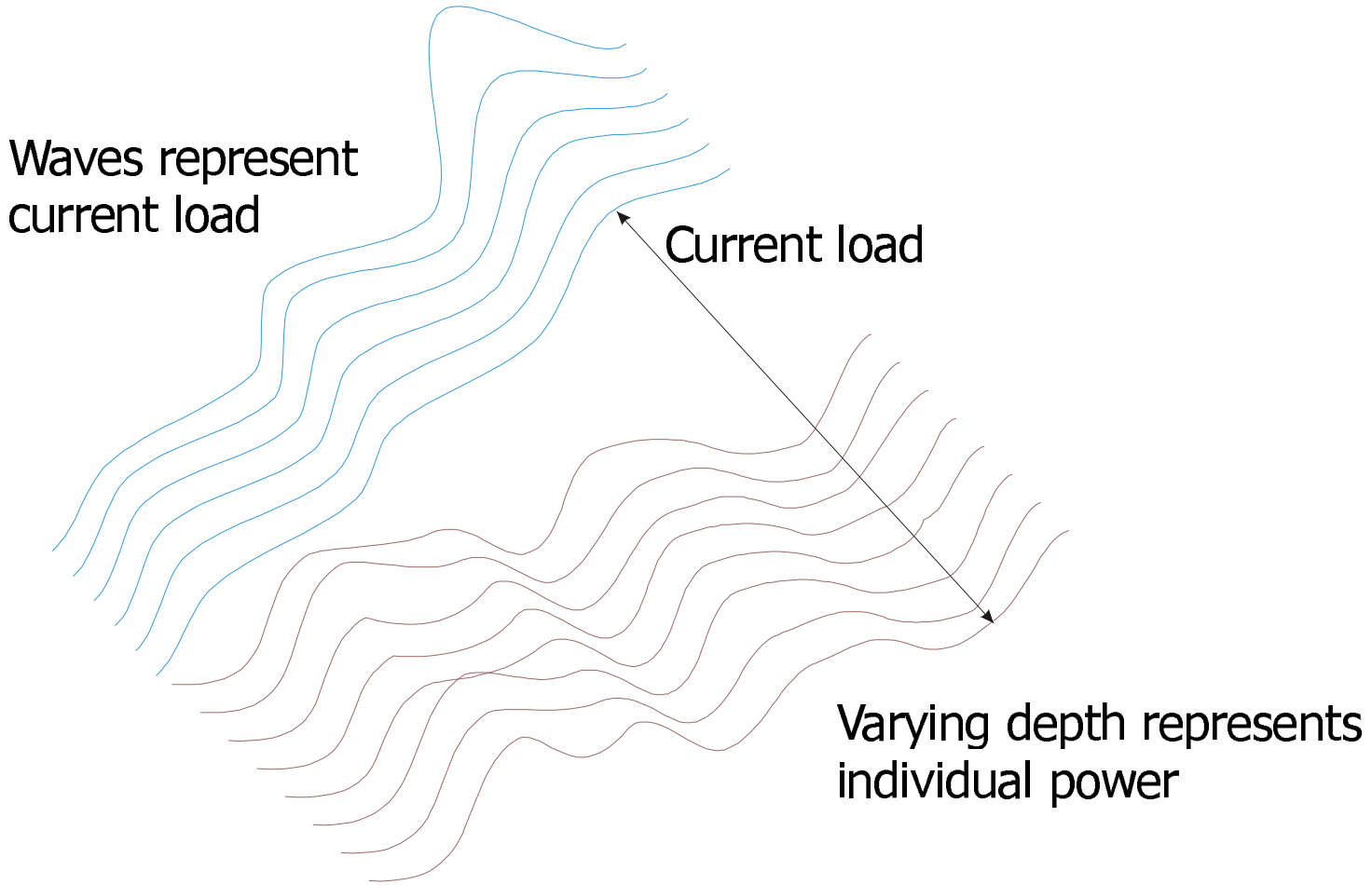
Model

- Code flows through the network as water in river basin to utilize available EEs
- Let's have surface made of *M* EEs
- Each EE is connected to several others EE
- Varying depth represents power of EEs
- Depth + wave height = current load
- Waves causes process migration
- Behavior of physical model is simulated
- Opened to utilization of applications like AVNMP

Model

- Topologies are discovered and mapped as the code flows
- Data processing while in transit
- Dispersion formula controls spreading of processes from EE
- Painting of left EEs
  => Advantage of receiver initiated algorithm
  for heavily loaded environments
- Forces of physical model have their counterparts
- Some tricks (e.g. painting) are presented

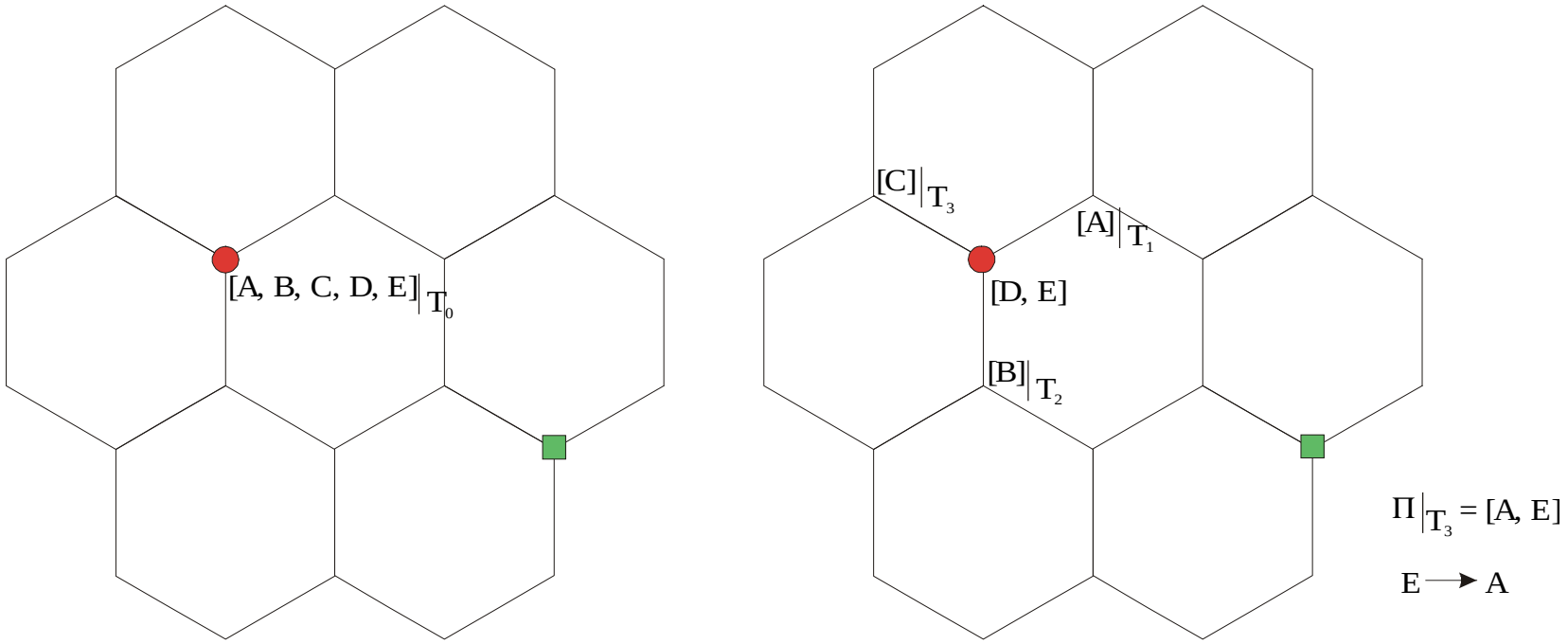
Model


Waves represent
current load
Current load
Varying depth represents
individual power

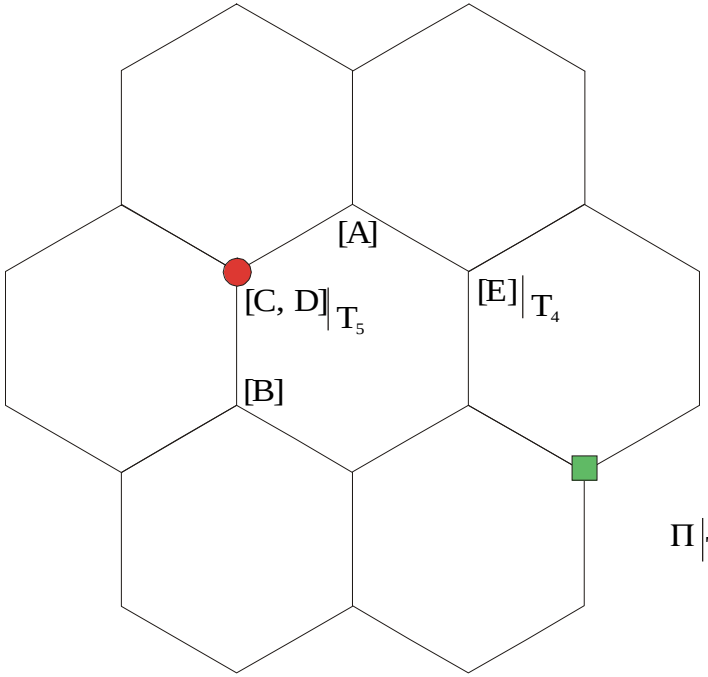
Distribution Algorithm

- EE is evaluated every time a process got created/moved there
- Process relocates if it is profitable or current EE will get overloaded too much
- Destination EE is selected in accordance with dispersion formula, left paintings, custom priorities and to
  - Minimize communication costs
  - Maximize execution speed
  - Move in desired direction when processing data in transit

Expected Behavior

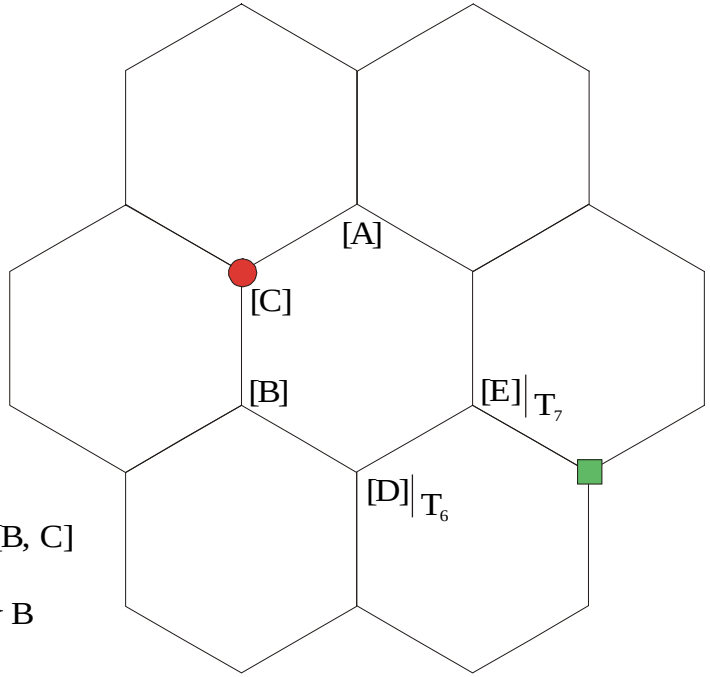


Expected Behavior



$\Pi|_{T_{4,5}} = [B, C]$

$C \rightarrow B$



$\Pi|_{T_6} = [B, C, D, E]$

$B \rightarrow C$

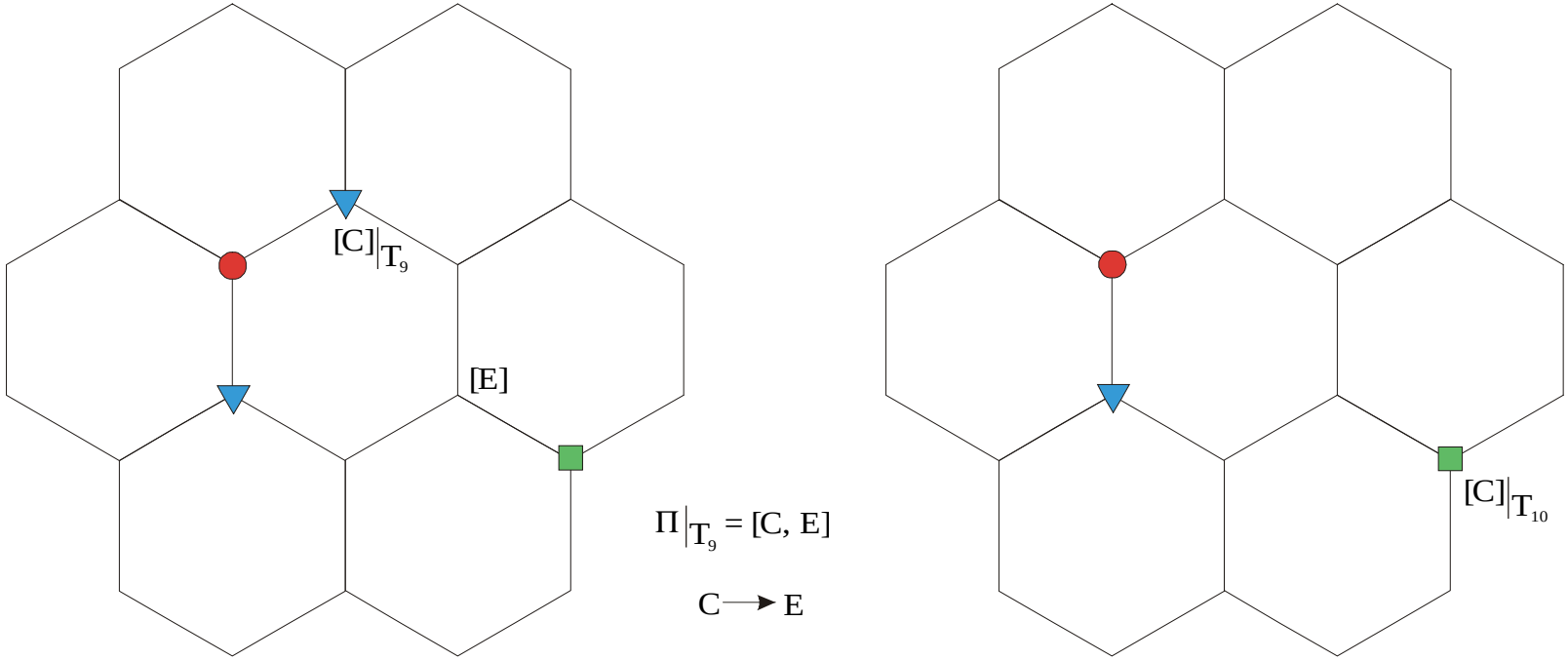
$E \rightarrow D$

$\Pi|_{T_7} = [A, B, C, D, E]$

$C \rightarrow A$

$B \rightarrow E \rightarrow D$

Expected Behavior



Comparison

- Found similar approach for decentralized load-balancing: The Particles Approach
- Different degree of fidelity with physical model
- Flat container vs. varying depth and waves
- Water vs. several non-mixable liquids
- Utilization of tricks
- Advantage of Active Networks
- Exploitation of environement‘s power

Future Work

- Completion of model and distribution algorithm
- MIT's ants framework
- Tuning and verifying of model and algorithm