Honeypot generator with network controllers and containerized infrastructure

Carol Sebastian Bontaș | Ioan-Mihail STAN | Răzvan Rughiniș
speaker

21st RoEduNet Conference
Networking in Education and Research

Context

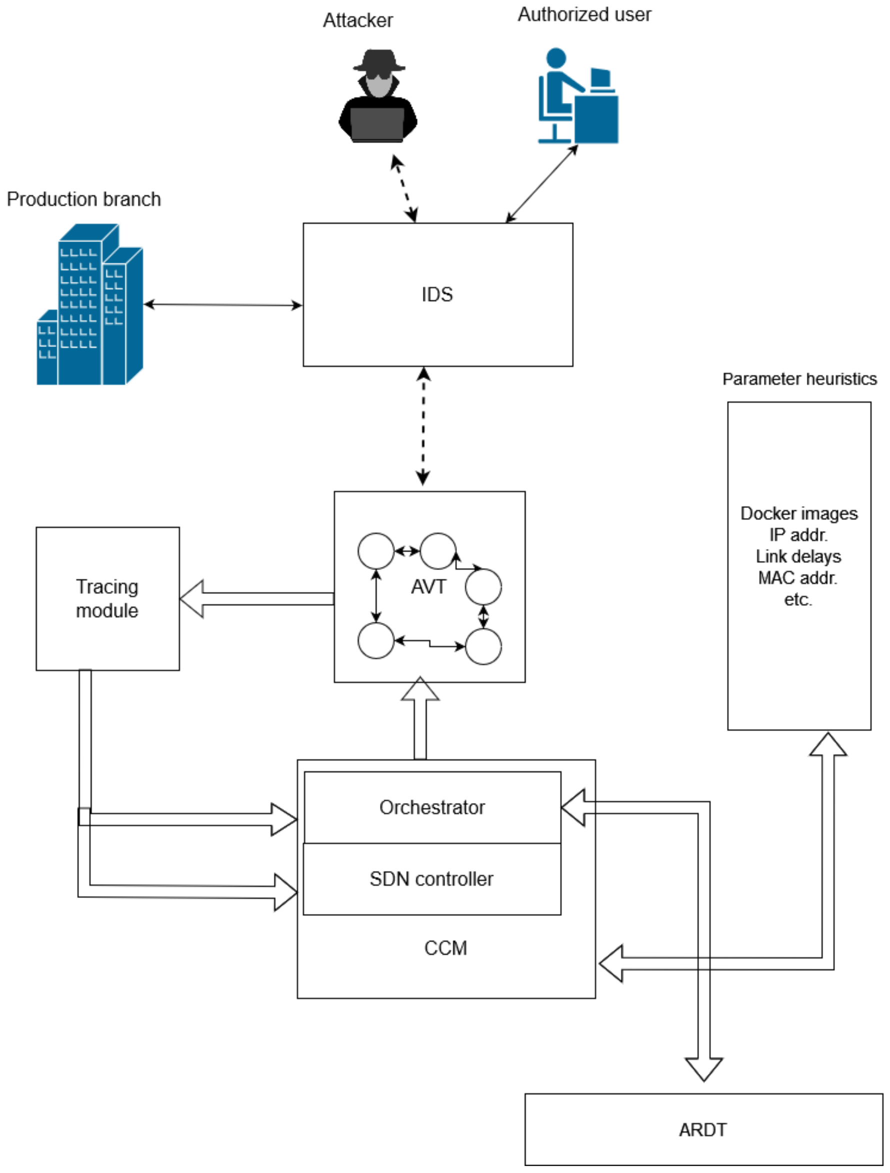
- Cyber crime global context requires complex behavior analysis
  - Q1 and Q2 2022 – over 8 mil data breaches
  - 93% of cases attackers can breach company networks
  - New or reborn methods of attacks

- Practices:
  - consistent security analysis of the assets exposed
  - low to high interaction honeypots

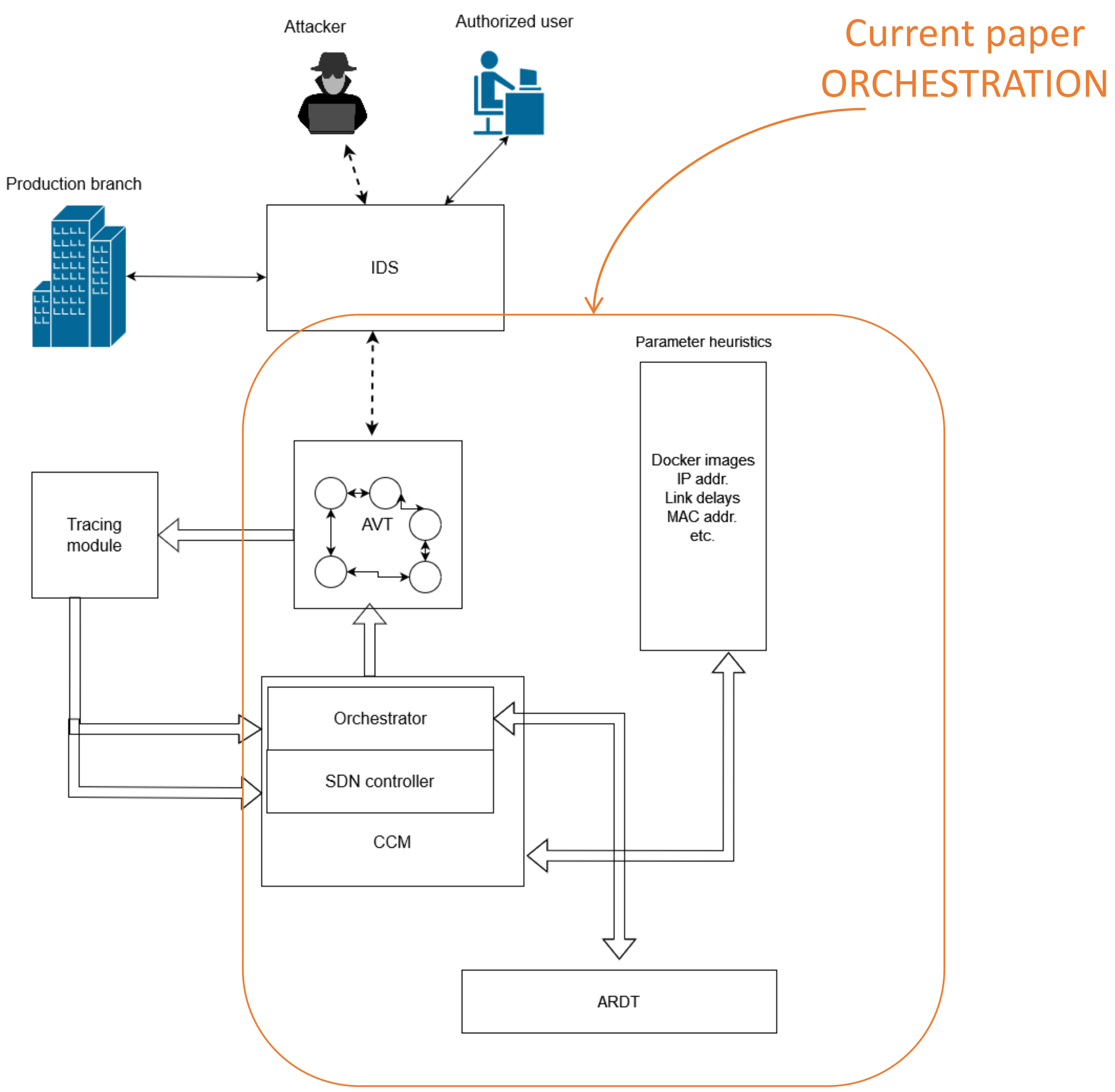
Problem statement

- Honeypot deployments follow similar configuration management recipes
  - They inherit a footprint of the company that manages the deployment in various infrastructures
  - High-Interaction honeypots require a degree of randomization
- Solution: Honeypot Generator
  - real time network manipulation with SDN controllers
  - virtual network and vulnerable service delivery with containerization engines

Concept Architecture



Concept Architecture

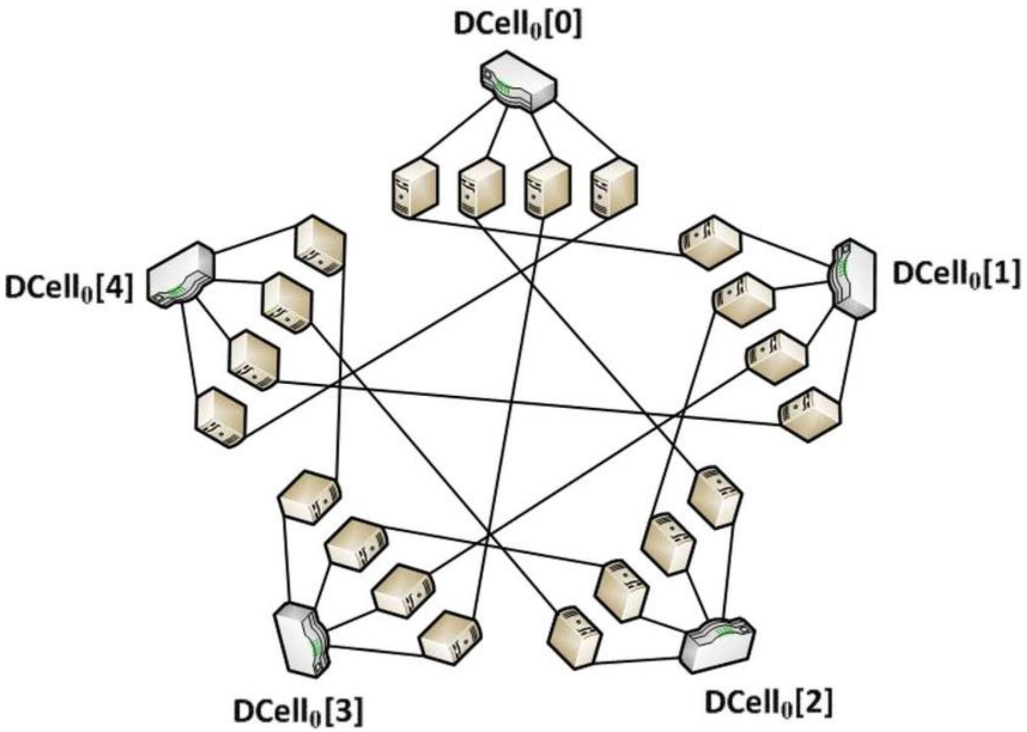


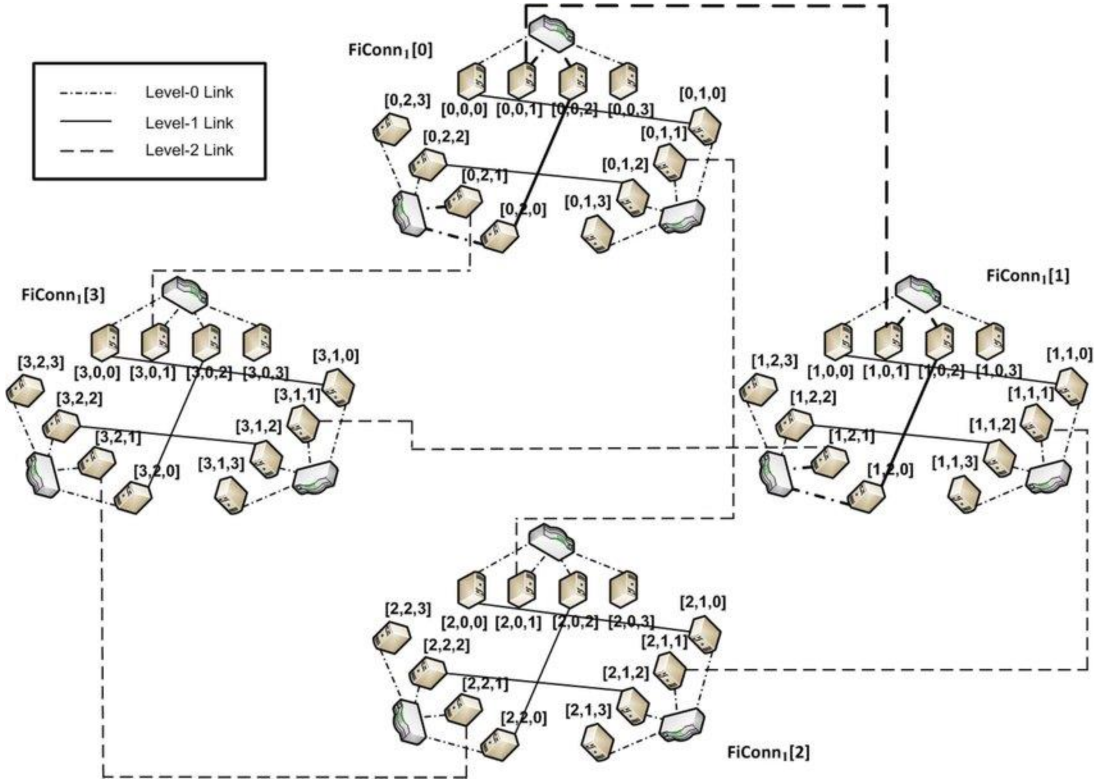
Honeypot Generator - orchestration

- Spins-up/Drills down neighbor containers, in real time, during the attack
- Uses Recursive Topologies heuristics to ensure scalability and predictability
  - E.g., DCell, FiConn
  - taken from the datacenter construction methodology and adapted
- Implements its own declarative language
  - Abstracts configuration routines
  - Enables limits and configuration particularities for the recursive communication topology

DCell and FiConn

Source of images: Li, Dan, et al. "Scalable and cost-effective interconnection of data-center servers using dual server ports." *IEEE/ACM Transactions on Networking* 19.1 (2010): 102-114.

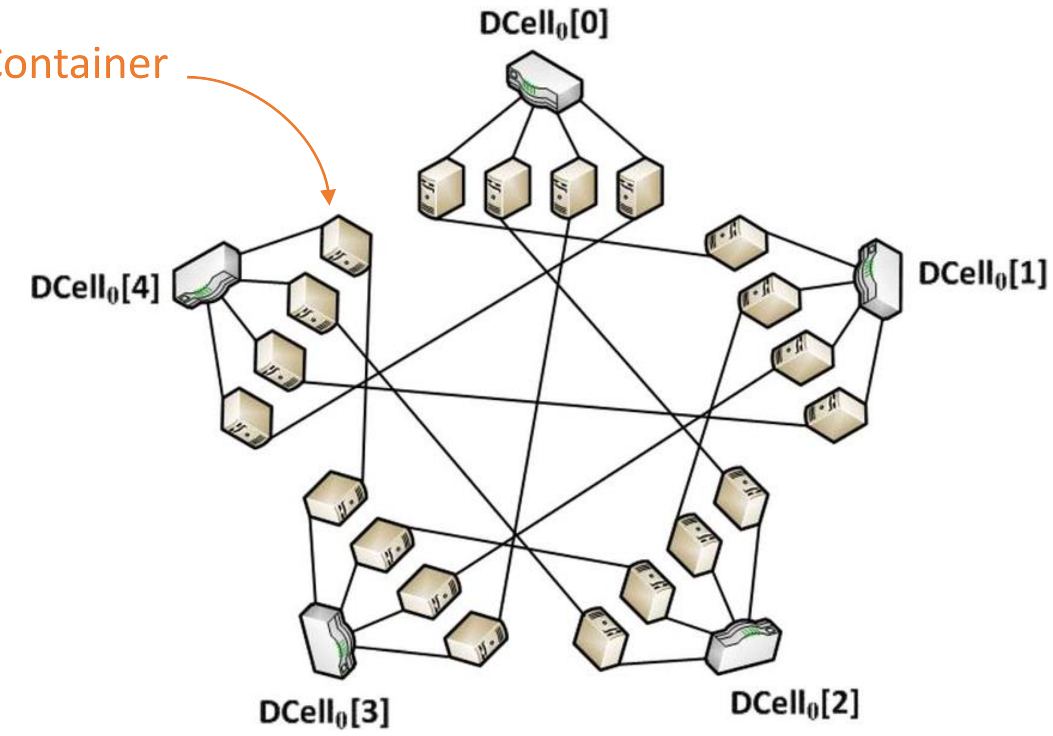


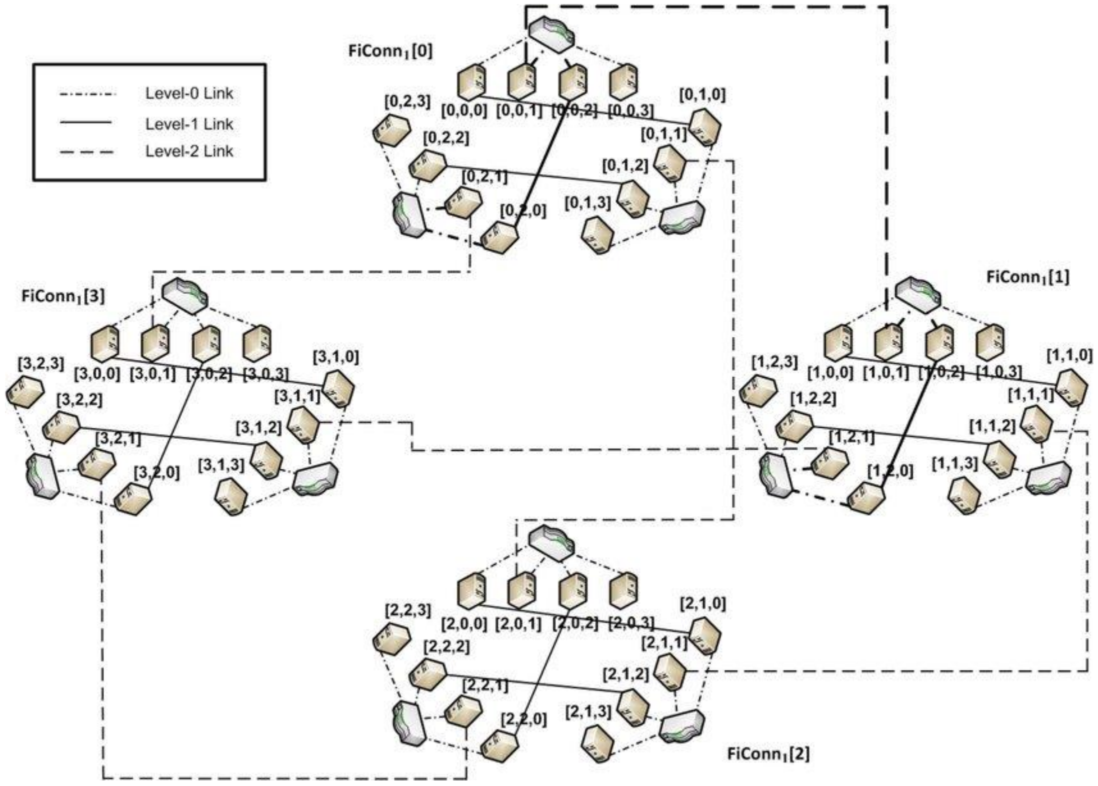


DCell and FiConn

Source of images: Li, Dan, et al. "Scalable and cost-effective interconnection of data-center servers using dual server ports." *IEEE/ACM Transactions on Networking* 19.1 (2010): 102-114.

Container





Configuration management - formalism

- Tuple N, K, f_c, c
- N - number of hosts found in a basic structure of grade 0
- K - maximum grade of the RDT
- c - maximum grade which imposes the same number and nature of the links for every node of grade c
- fc- function which states how many links of every grade a node of grade c must have

Configuration management - pragmatically

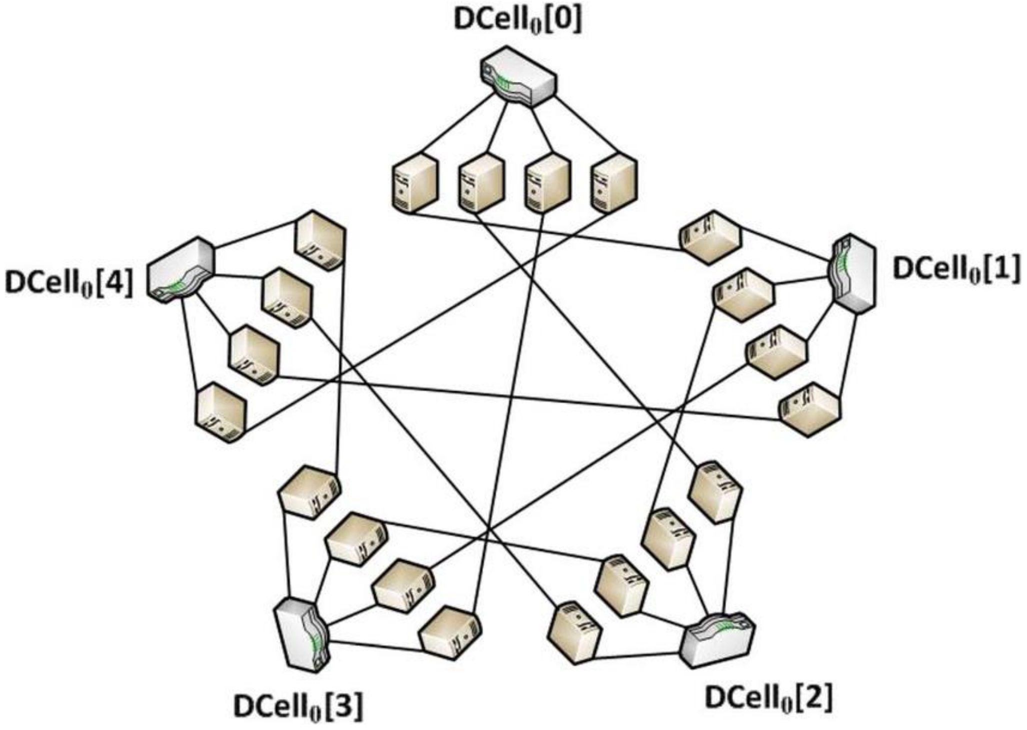
N, K, f_c, c

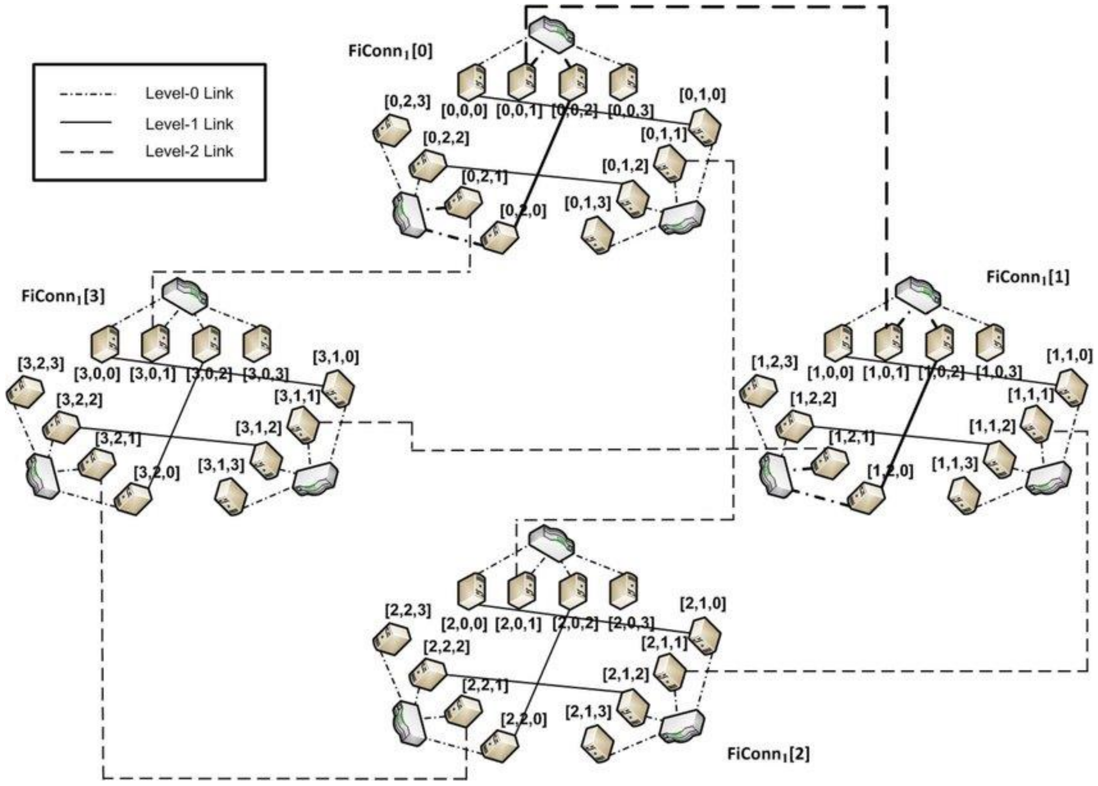
```
{
    "constraints" :
    {
        "grade" : 0,
        "links" :
        {
            "0" : 1,
            "1" : 1,
            "2" : 1,
            "3" : 1,
            "4" : 1
        },

        "numHosts" : 2
    },

    "debug":
    {
        "linkHistory" : true
    },

    "optimizations":
    {
        "enableCollapse" : true,
        "enablePrunePause" : true
    }
}
```

```
route entry 10.0.0.1 network 10.0.0.0/8
route 10.0.0.1 link 10.0.0.0/24
route 10.0.0.1 add net 10.4.0.0/14 tunnel "thishost" "honeyd-b"
route 10.0.0.1 add net 10.1.0.0/16 10.1.0.1 latency 55ms loss 0.1
route 10.0.0.1 add net 10.2.0.0/16 10.2.0.1 latency 20ms loss 0.1
route 10.0.0.1 add net 10.3.0.0/16 10.2.0.1 latency 20ms loss 0.1
route 10.1.0.1 link 10.1.0.0/24
route 10.2.0.1 link 10.2.0.0/24
[...]
route 10.2.0.1 add net 10.3.0.0/16 10.3.0.1 latency 10ms loss 0.1
route 10.3.0.1 link 10.3.0.0/24
route 10.3.0.1 add net 10.3.1.1/24 10.3.1.1 latency 10ms
route 10.3.0.1 add net 10.3.240.0/20 10.3.240.1 latency 5ms
route 10.3.1.1 link 10.3.1.1/24
route 10.3.240.1 link 10.3.240.0/20
route 10.3.240.1 add net 0.0.0.0/0 10.3.0.1 latency 40ms loss 0.5
[...]
bind 10.2.0.243 to fxp0
bind 10.3.1.15 to fxp0
```

Configuration management – pragmatically(2)

Source of images: Li, Dan, et al. "Scalable and cost-effective interconnection of data-center servers using dual server ports." *IEEE/ACM Transactions on Networking* 19.1 (2010): 102-114.





$N = 4, K = 1, f_c(x) = 1, c = 0$

$N = 4, K = 3, f_c(1) = 2, f_c(2) = 3, c = 1$

Control Plane Algorithm

1. Calculate $H(x)$=$number\ of\ nodes\ of\ grade\ x\ which\ make\ a\ node\ of\ grade\ x$+1

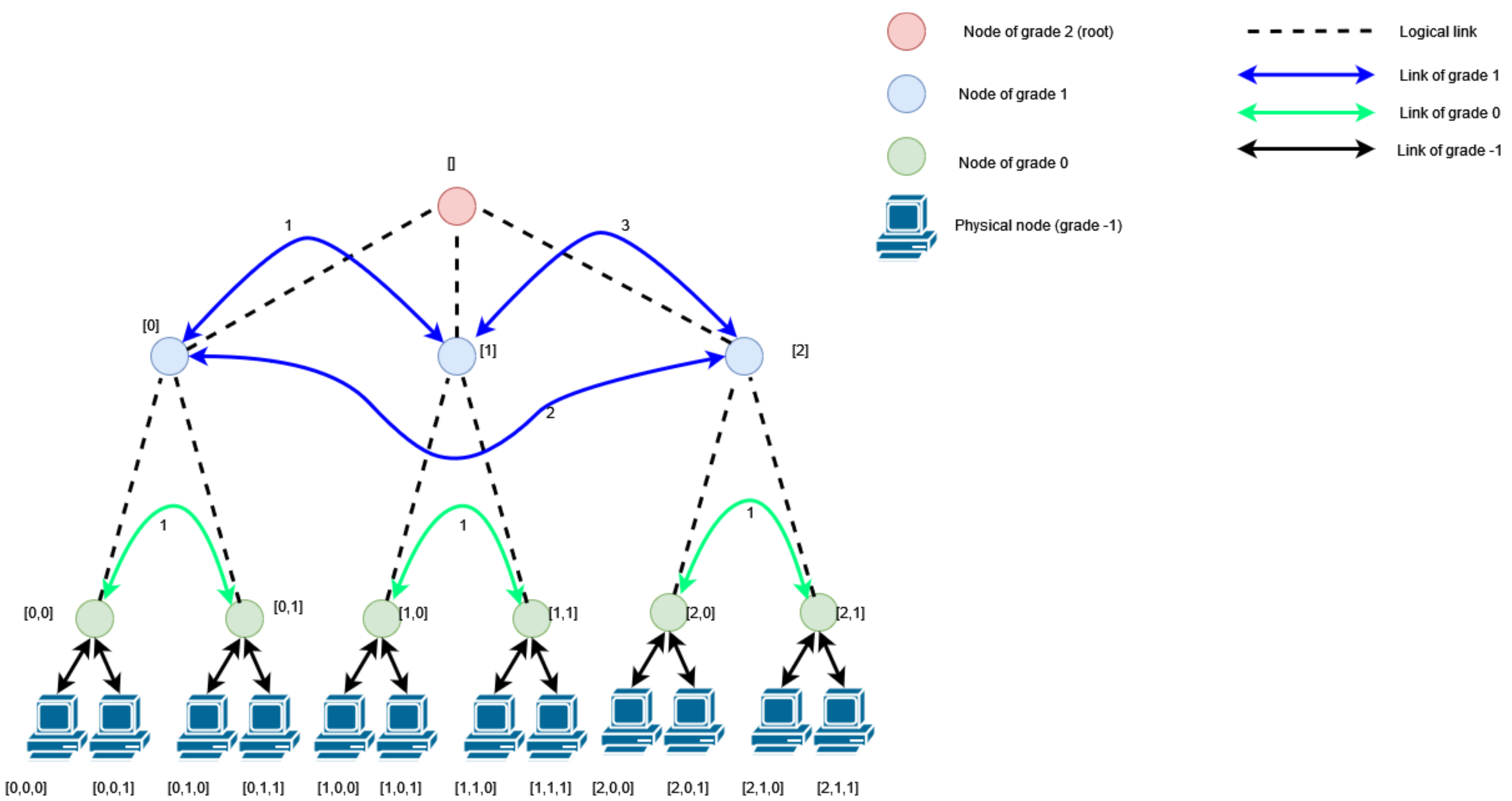
$$H(x) = f_c(x-1)\prod_{x-1}^{i=c} H(i) + 1$$
$$H(0) = N$$

2. Initialize linked tree structure where root is the node of grade K and every node of grade x + 1 has H(x) children

3. Every higher-grade logical node equally distributes its higher-grade links to its children until we reach nodes of grade c

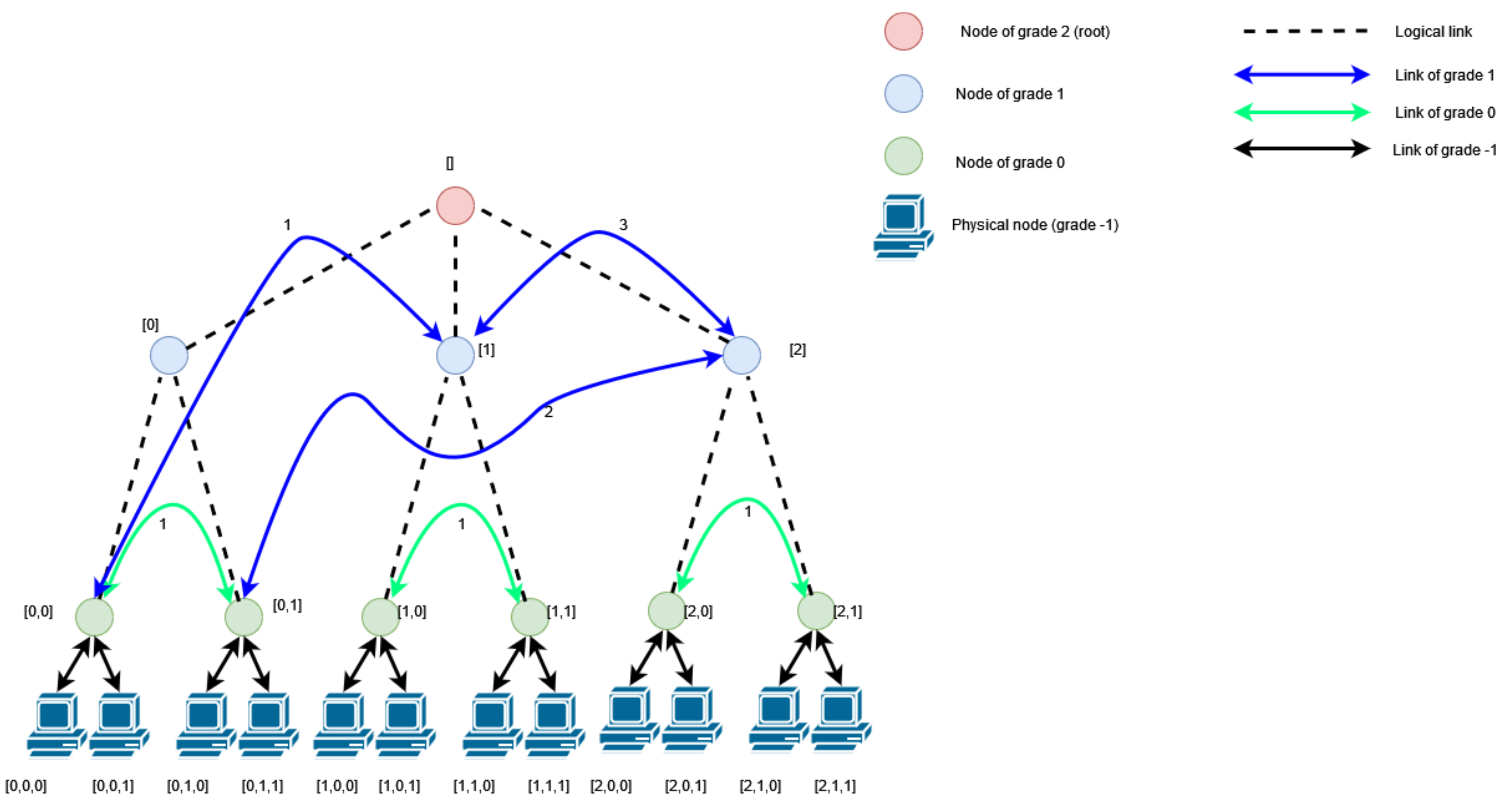
4. If c is higher than 0, then we wait for a user heuristic to distribute the links until they cover all containers. Default, it always gives to the first container in a cell all the links of the parent cell

Control Plane Algorithm – Initialization
simplified



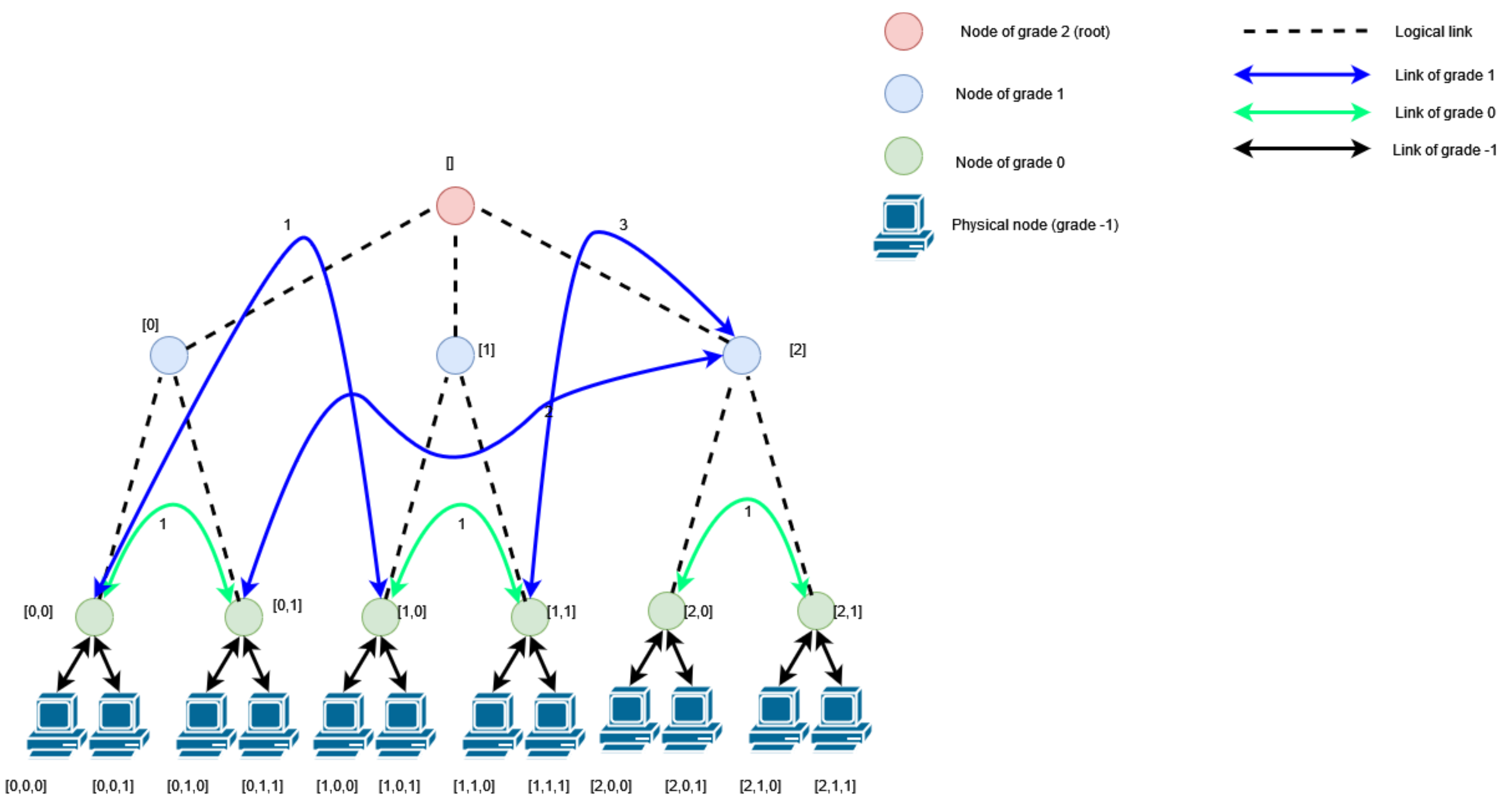
Control Plane Algorithm – Propagation

simplified



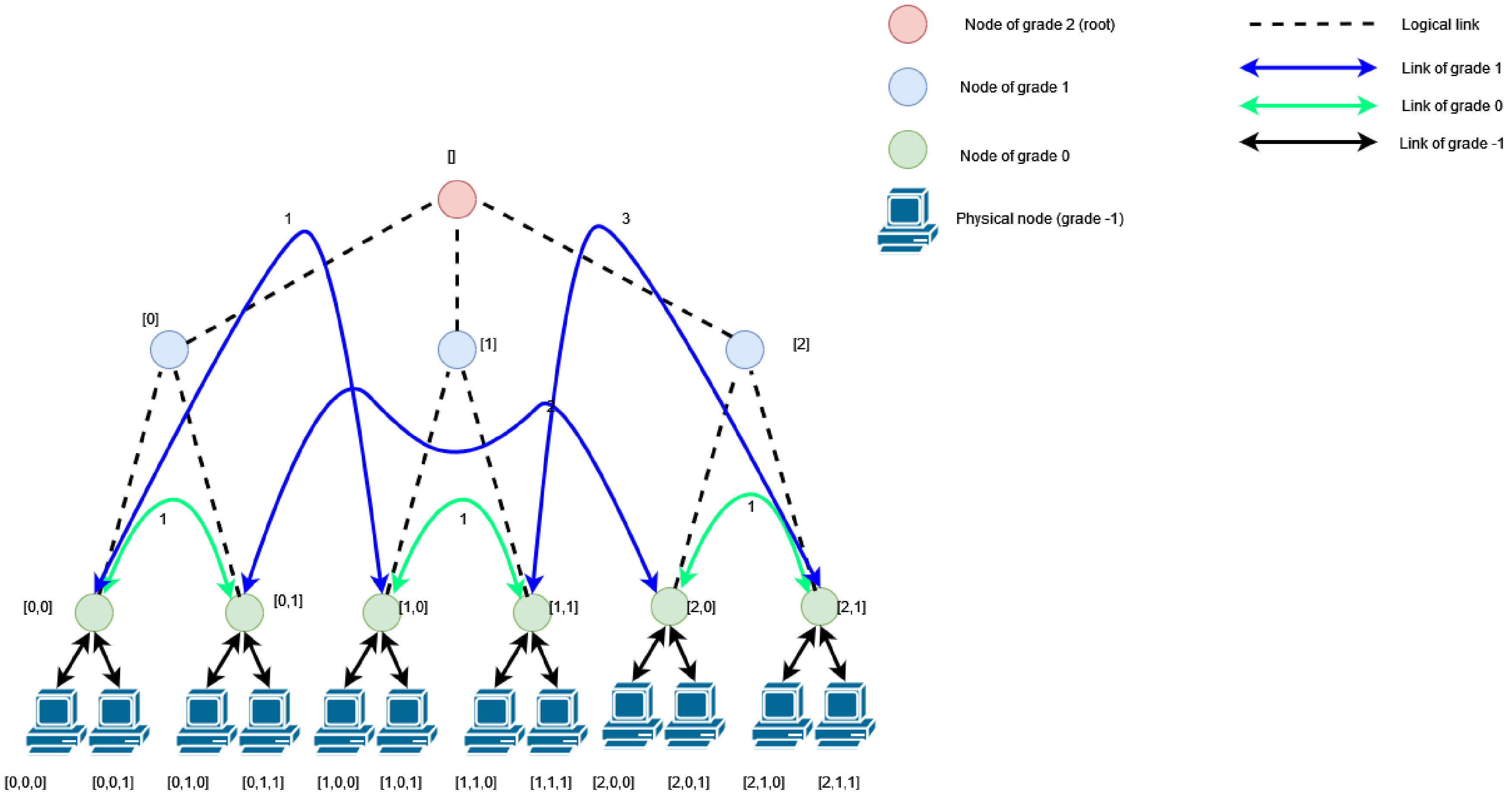
Control Plane Algorithm – Propagation

simplified



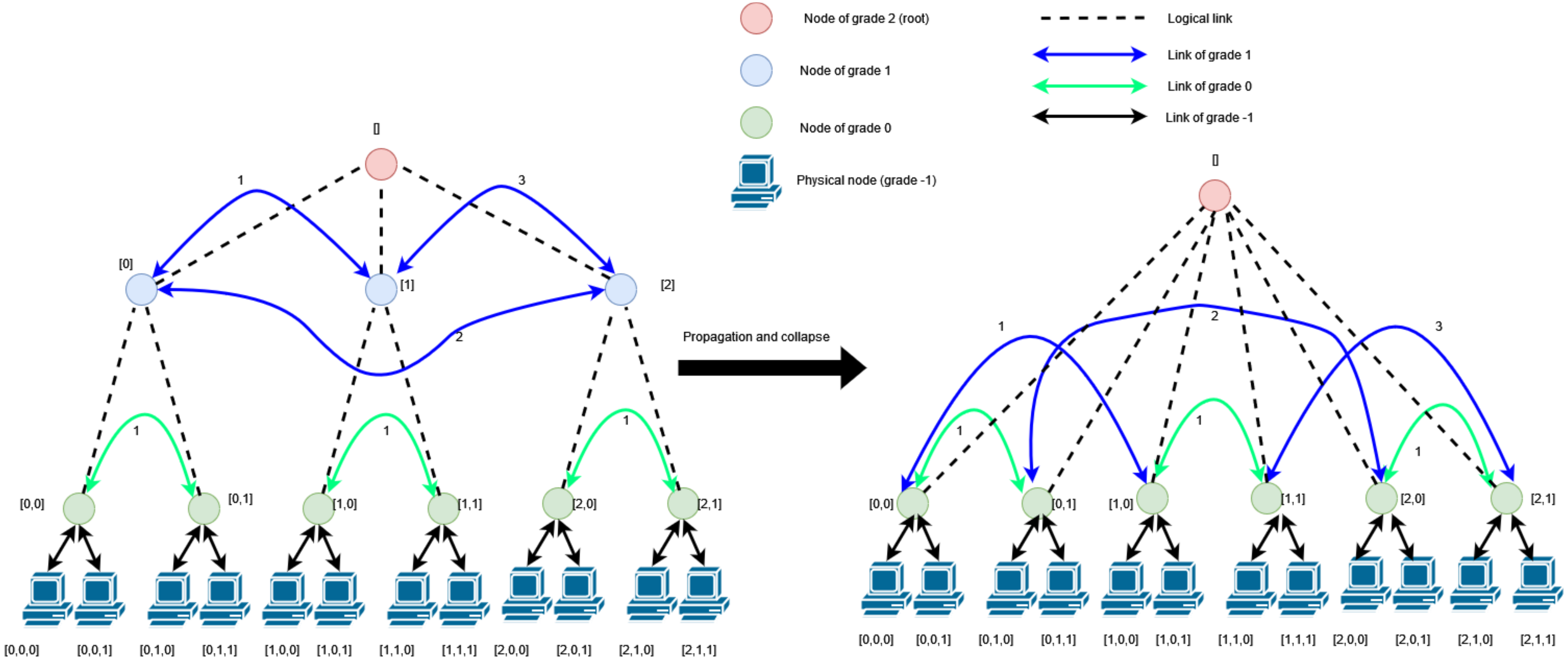
Control Plane Algorithm – Propagation

simplified



Control Plane Algorithm – Collapse

simplified



Conclusions

- Developed the control plane of a Honeypot Generator
  - Smart orchestration
  - Adaption of Recursive Defined Topology algorithms for vulnerable container orchestration
- Created a declarative language to propagate the configuration decisions
  - Tuple N, K, f_c, c
  - One can defined DCell or FiConn *-like* topologies using the config parameters
- Provided an architecture for a high-interaction honeypot with real-time orchestration using emerging technologies:
  - SDN controllers for traffic forwarding
  - Virtual Switches
  - Containers
  - Recursive Defined Topologies for scalability in time