**Simulation**

Meeting in Kleingruppe: 09.05.2016

**Klassendiagramm => Controller**

1. SimulationsManager:
   1. Wie kann eine Simulation gestartet werden?

3 verschiedene Möglichkeiten:

* + 1. „normale“ Simulation
    2. erweiterte Simulation
    3. über Testszenario
  1. zusätzliche Funktionen:
     1. **createSimulation()**
        1. Parameter:
           1. IP für Source
           2. IP für Destination
           3. TTL
           4. Tags: => Zusatzparameter: ob z.B. verschlüsselt oder nicht
     2. **createPacket()**
        1. Parameter:
           1. ~~IP für Source~~ => Hardwarenode
           2. ~~IP für Destination~~ => Hardwarenode
           3. TTL
           4. Tags:

Zusatzparameter: ob z.B. verschlüsselt oder nicht

* + - 1. Rückgabe: erzeugtes Packet

1. Simulation => im Klassendiagramm: Model
   1. **addPacketSend(packet)**
      1. Hinzufügen eines Paketes in die packetSend – Liste
   2. **simulation()**
      1. Parameter:
         1. Id: String
         2. expected result: bool
   3. **execute()**
      1. startet für jedes Packet
2. Packet => im Klassendiagramm: Model
   1. **packet()**
      1. Parameter:
         1. Hardwareknoten als Source
         2. Hardwareknoten als Destination
         3. TTL
         4. Tags
      2. Diese Funktion soll Source in Hops-Liste eintragen
   2. **send()**
      1. Parameter:
         1. keine
      2. Rückgabewert: packet
         1. Wenn 0, dann Fehler. Ansonsten ist es das Rückpacket.
   3. neue Attribute in der Klasse selber:
      1. **string result**
3. Networkmanager => im Klassendiagramm: Controller
   1. **getWorkstationByIp()**
      1. Parameter:
         1. IP
      2. Rückgabe: ~~array =>~~ Hardwarenode
      3. wird für Start- und Destination aufgerufen
      4. Hinweis: Broadcast, Multicast
4. Hardwarenode => im Klassendiagramm: Model
   1. **send()**
      1. Parameter:
         1. destination (ist ein Hardwarenode)
         2. tags
      2. Rückgabe:
         1. Hardwarenode
            1. 0, wenn Fehler (hat nicht geklappt) ansonsten ist es der nächste Hardwareknoten (der, an den es gesendet wurde).
   2. **receive()**
      1. Parameter:
         1. tags
      2. Rückgabe:
         1. bool
      3. Hardwareknoten empfangen
5. Interface ILayer => im Klassendiagramm: Model
   1. **validate()**
      1. Parameter: können sich noch ändern, abhängig davon welche Schichten benötigt werden
         1. connections
         2. routingtable
         3. interfaces
      2. überprüft ob Verbindung da ist, Zertifikat gültig ist,…
         1. Frage an die Gruppe: Welche Prüfungen sollen genau gemacht werden? Was macht Rechnernetz da genau?
   2. **getNextHardwareNode() => diese Methode löschen!!!**