Network Algorithm – EP2400
Assignment 2

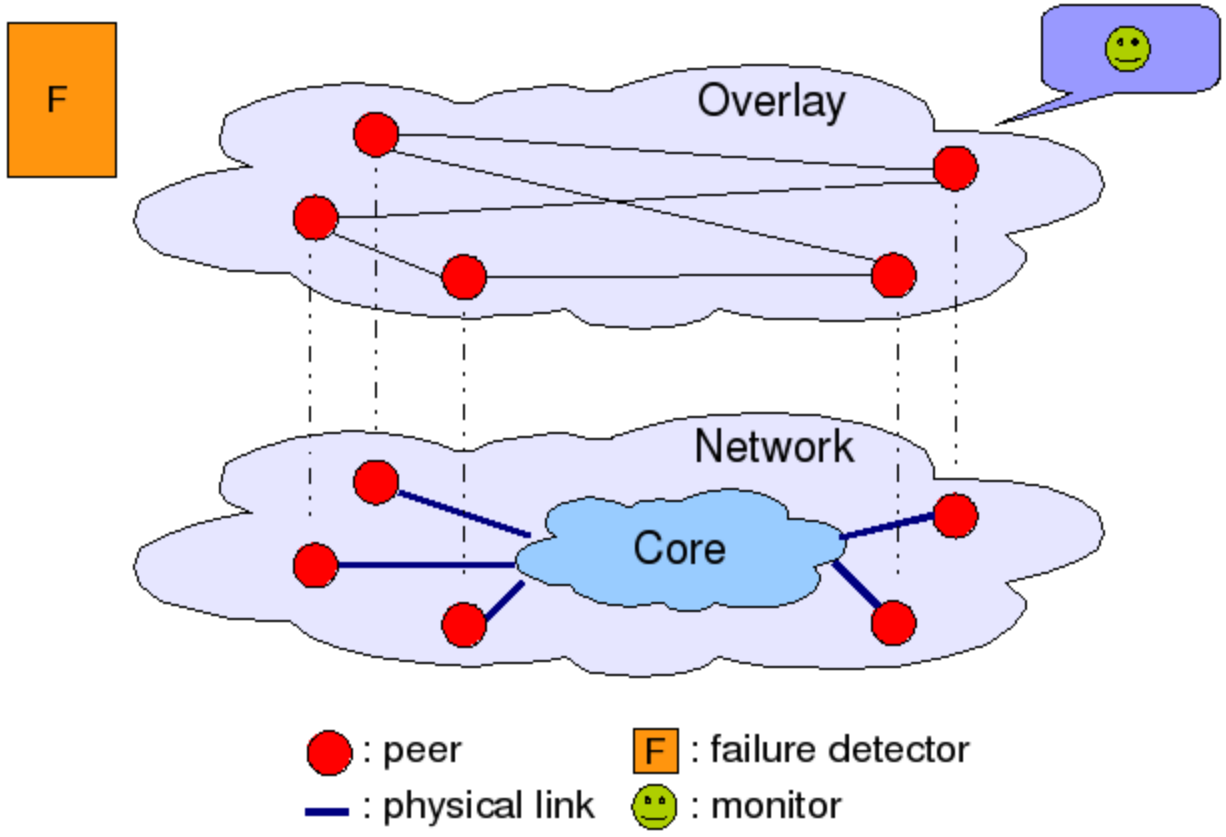
Amir H. Payberah
amir@sics.se


SICSIM

SICSIM

- SICSIM is a Discrete Event, Flow Level simulator for simulating Peer-to-Peer overlays.
- Discrete event model:
  - The system state may change only at discrete point in time.
- Flow level model:
  - It does not generate events for each packet arrival or departure.
  - Events are generated only when the rate of flows change.
    - Sending a message
    - Receiving a message
    - ...

SICSIM – Main Modules



Peer

- Main methods of peer:
  - create
  - join
  - receive
  - failure
  - leave
  - sendMsg
  - loopback
  - signal

Monitor

- Main methods of monitor:
  - verify
  - snapshot
  - update

Failure Detector

- Main methods of failure detector:
  - register
  - unregister

Overlay

- Main methods of overlay:
  - add
  - remove
  - getRandomNodeIdFromNetwork
- A peer can only see the peers who have joined the overlay.

SICSIM - Configuration

- There are a few variables to configure SICSIM.
- These variables are defined in a text file: sicsim.conf.

How to define the behaviour of system?

Scenario

- Scenario defines the way that the simulator behave.
- Scenario is defined in a text file.
- The scenario file can have one scenario or multiple scenarios:
  - Lottery event
  - Delay event
  - Monitor event
  - Signal event
  - ...

Lottery Event

```
type:       lottery
peer:       sicsim.samples.helloworld.Peer
link:       sicsim.network.links.ReliableLink
count:      100
interval:   20
join:       7
leave:      2
failure:    1
```

Delay Event

```
type:    delay
delay:   1000
```

Monitor Event

```
type:      monitor
monitor:   sicsim.samples.helloworld.OverlayMonitor
```

Signal Event

```
type:       signal
count:      12
interval:   10
signal:     5
```

Sample Scenario

```
### Scenario 1 ###
type:                monitor
monitor:             sicsim.dht.chord.DHTMonitor
---
### Scenario 2 ###
type:                lottery
peer:                sicsim.dht.chord.Peer
link:                sicsim.network.links.ReliableLink
count:               512
interval:            10
join:                1
leave:               0
failure:             0
---
### Scenario 3 ###
type:                delay
delay:               5000
---
### Scenario 4 ###
type:                signal
count:               512
interval:            1
signal:              1
---
### Scenario 5 ###
type:                lottery
peer:                sicsim.dht.chord.Peer
link:                sicsim.network.links.ReliableLink
count:               100000
interval:            10
join:                1
leave:               0
failure:             1
```

Project Outline

Project Outline

- You will be required to compare Chord, Chord# and Kademlia in different network setting and evaluate some metrics.

Network Setting

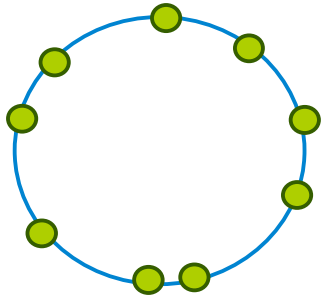
- Network size
- Churn level
- Identifier distribution

Network Setting – Churn level

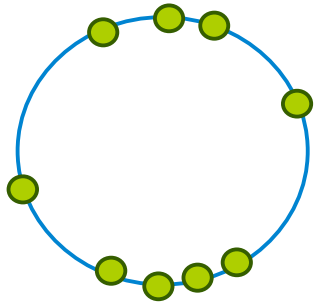
- No churn
  - The network size is static i.e. no joins and failures.
- Low churn
  - Peers join and fail continuously and the mean inter-arrival time of events is 10 time units.
- High churn
  - Peers join and fail continuously and the mean inter-arrival time of events is 5 time units.

Network Setting – Identifier distribution

- Uniform distribution
  - The nodes' identifiers are distributed uniformly over the identifier range.



- Skewed distribution
  - The identifier range is divided into a number of clusters and the peers are placed in these clusters with a user defined probability



Metrics

- Bandwidth consumption
  - The number of maintenance messages sent by each peer.
- Lookup forward load
  - The number of lookup messages that each peer has to forward.
- Lookup hops
  - The number of hops that a lookup message passes through to find the result.
- Failed lookup
  - The number of unsuccessful lookups.

Typical Experiment

- A typical experiment comprises of three sequential parts:
  - Network creation
  - Network stabilization
  - Taking measurements

Network creation

Network stabilization

Taking measurements

```
### Scenario 1 ###
type:                monitor
monitor:             sicsim.dht.kademlia.DHTMonitor
---
### Scenario 2 ###
type:                lottery
peer:                sicsim.dht.kademlia.Peer
link:                sicsim.network.links.ReliableLink
count:               512
interval:            10
join:                1
leave:               0
failure:             0
---
### Scenario 3 ###
type:                delay
delay:               5000
---
### Scenario 4 ###
type:                signal
count:               512
interval:            1
signal:              1
```

Source Code

- sicsim.dht.chord
- sicsim.dht.chordsharp
- sicsim.dht.kademlia

Source Code

- Each package contains two classes:
  - Peer: a class for implementing the behaviour of a peer.
  - DHTMonitor: a class to verify the structure of DHT.

Your Tasks ...

Tasks

- Task 1
  - Implementing Chord, Chord# and Kademlia.
- Task 2
  - Analyzing the behaviour of each DHT in different configurations.
- Task 3
  - Comparing the implemented DHTs together in different scenarios.

Task 1

Task 1

- Implementing Chord, Chord# and Kademlia.

Task 1 – Chord and Chord#

- Important variables:
  - succ
  - pred
  - fingers
  - succList
- DHTMonitor uses these variables to verify the structure of Chord and Chord#.
  - The verify method of DHTMonitor

Task 1 – Chord and Chord#

- The peers can use failure detector for their successor and predecessor.

- Useful methods:
  - belongsTo: id ∈ (start, end]
  - belongsTonn: id ∈ (start, end)
  - ...

Task 1 – Kademlia

- Important variables:
  - kbucketList
- DHTMonitor uses this variable to verify the structure of Kademlia.
  - The verify method of DHTMonitor
- The peers can not use failure detector.

Task 2

Task 2

- Analyze the behaviour of each DHT in different configurations.
  - There are some parameters in each DHT that influence the behaviour of that DHT.
- Draw a plot for each measurement.
- System setting:
  - Network size: 512 nodes.
  - Churn level: Low churn.
  - Identifier distribution: Uniform distribution.

Task 2 – Chord and Chord#

- Important parameters:
  - Successor list size
  - Successor stabilization interval
  - Finger list stabilization interval

- Assume:
  - Successor list size = 9.
  - The successor stabilization and the finger list stabilization are synchronized, and they are called together.
  - Set the stabilization interval to 20, 50, 80 time unit.

Task 2 – Chord and Chord#

- Measure the following metrics:
  - Bandwidth consumption
    - X-axis: The stabilization interval.
    - Y-axis: Total bandwidth consumption in all peers.
  - Failed lookups
    - X-axis: The stabilization interval.
    - Y-axis: The average failed lookup during the simulation time (ratio of failed lookup to the all lookups).

Task 2 – Kademlia

- Important parameters:
  - Nodes per entry (k)
  - Parallel lookups (α)
  - Number of ID returned
  - Stabilization interval

- Assume:
  - k = 5.
  - Each peer returns 5 IDs in replay.
  - The stabilization interval is 200 time unit.
  - Set the α to 1, 3 and 5.

Task 2 – Kademlia

- Measure the following metrics:
  - Lookup hops
    - X-axis: The values of α.
    - Y-axis: The average lookup hops during the simulation time.
  - Failed lookups
    - X-axis: The values of α.
    - Y-axis: The average failed lookup during the simulation time (ratio of failed lookup to the all lookups).

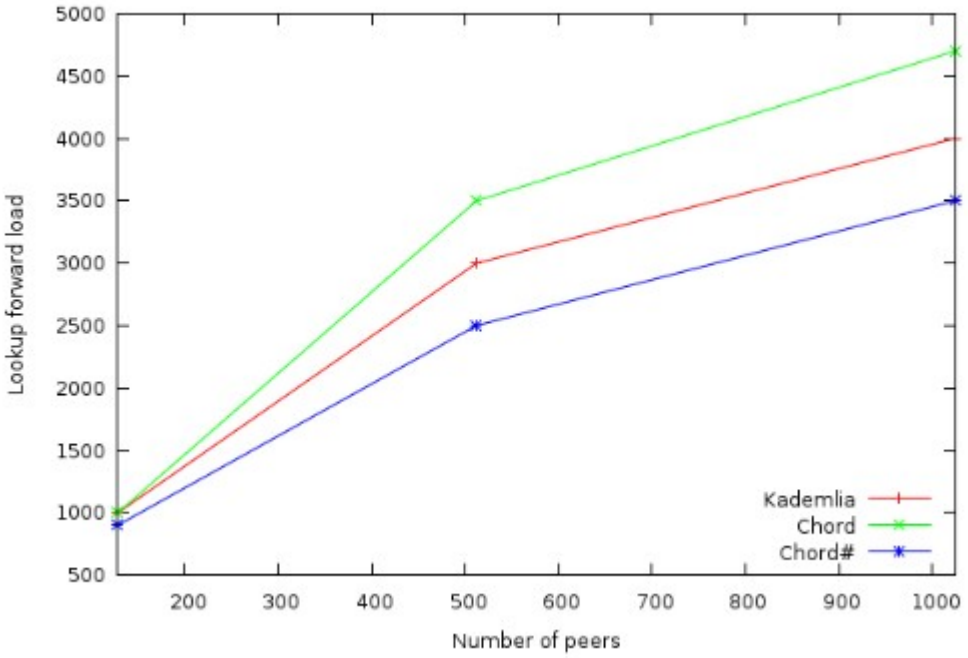
Task 3

Task 3

- Compare these DHTs together in different network setting.
- Chord and Chord# configuration
  - Successor list size = $\log_2 N$ (N: number of nodes in system)
  - Successor stabilization interval = 50 time unit
  - Finger list stabilization interval = 50 time unit
- Kademlia configuration
  - k = 5
  - α = 3
  - Number of ID returned = 5
  - Stabilization interval = 200 time unit

Task 3

- Draw a plot for each measurement, such that contains the information of Chord, Chord# and Kademlia.



Task 3 – Setting 1

- The system setting:
  - Network size: 128, 512 and 1024.
  - Churn level: No churn.
  - Identifier distribution: Uniform distribution.

Task 3 – Setting 1

- Measure the following metrics:
  - Lookup forward load
    - X-axis: The number of peers in the system.
    - Y-axis: The average lookup forward load.
  - Lookup hops
    - X-axis: The number of peers in the system.
    - Y-axis: The average number of lookup hops for each DHT.

Task 3 – Setting 2

- The system setting:
  - Network size: 512.
  - Churn level: No churn.
  - Identifier distribution: Uniform distribution and skewed distribution.

Task 3 – Setting 2

- Measure the following metrics:
  - Lookup forward load (for each identifier distribution)
    - X-axis: The lookup forward load.
    - Y-axis: The number of peers.
  - Lookup hops
    - X-axis: The identifier distribution.
    - Y-axis: The average lookup hops.

Task 3 – Setting 3

- The system setting:
  - Network size: 512.
  - Churn level: Low churn and high churn.
  - Identifier distribution: Uniform distribution.

Task 3 – Setting 3

- Measure the following metrics:
  - Lookup hops
    - X-axis: The churn levels.
    - Y-axis: The average number of lookup hops for each DHT.
  - Failed lookup
    - X-axis: The churn levels.
    - Y-axis: The average number of failed lookups for each DHT (ratio of failed lookup to the all lookups).

How to do the measurements?

Gathering Statistics

- Class Peer provides some variables to gather statistics:
  - FailedLookup
    - A static variable that counts all the failed lookups
  - HopCount
    - A static variable that shows the total hop counts in all lookups.
  - BwConsumption
    - A static variable that measures the traffics generated from all peers
  - LookupForwardLoad
    - A local variable that keeps the lookup traffic in each peer

Gathering Statistics

- The peers gather statistics when the first peer receives a signal from the simulator.
- The simulator sends signal to peers by calling the signal method of Peer.

```
type:       signal
count:      12
interval:   10
signal:     5
```

Lookup Service

- Implement the signal method, such that the peer performs a lookup for a random key.
- The lookup service should be called periodically.
  - The interval = 50 time unit.

Failed Lookup

- Time out
  - If the peer does not receive the reply of the key lookup.
  - Before starting the next lookup, each peer should check whether they have received the reply of previous lookup or not.

- Wrong reply
  - If the reply of the key lookup is wrong.
  - If they have received the reply, they should ask simulator about the correctness of the result.
  - ((DHTMonitor)this.monitor).checkLookup(host, id).

Last Page

- Before start the assignment:
  - Read the simulator document carefully ...
  - Read the assignment document completely ...

Question?