NS2: Contents

- NS2 – Introduction to NS2 simulator
- Some NS2 examples
- NS2 project work instructions

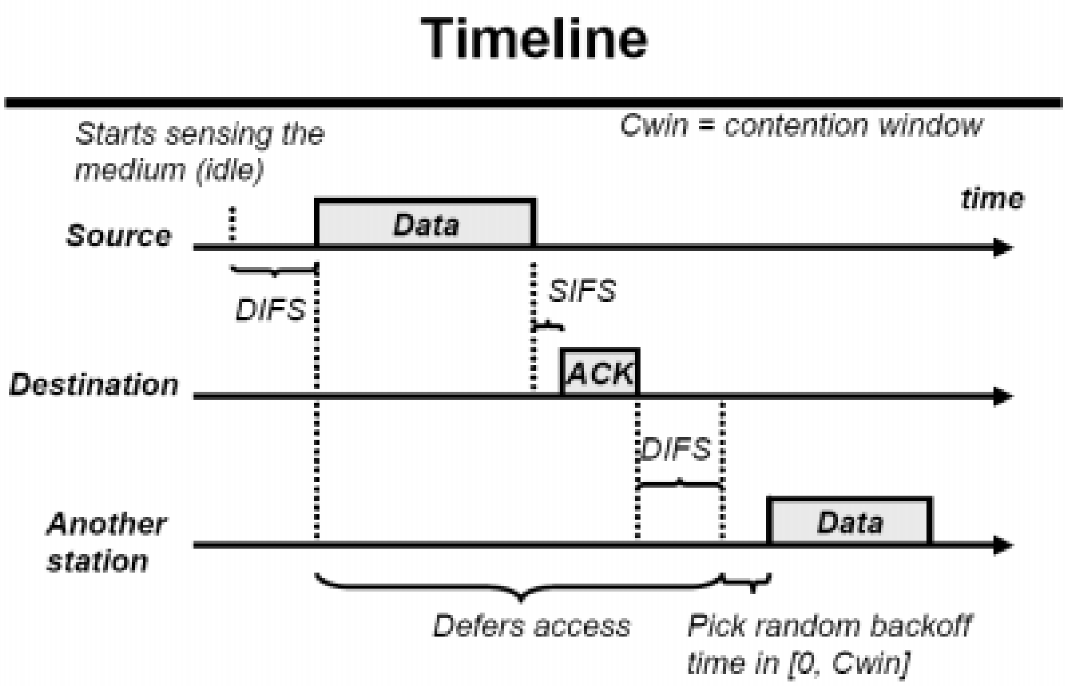
Introduction

- The ns2 assignment is about
  - 802.11 DCF MAC mechanism and
  - its interaction with higher layer protocols (UDP/TCP)
- Two traffic scenarios
  - Static:

802.11 MAC layer features

- 802.11 standard specifies two MAC mechanisms
  - PCF (Point Coordination Function): base station polls clients, not used in practice
  - DCF (Distributed Coordination Function): random access scheme, normally used in WLAN networks
- DCF features
  - Random access based on carrier sensing with guard intervals
  - Smaller guard intervals in channel access for small control packets (prioritized traffic)
  - Packets are acknowledged at the link layer and retransmitted in no ACK is received
    - Exponential backoff
  - In wireless multihop networks additional problems occur due to hidden/exposed nodes
    - RTS/CTS handshake before data transmission
- In our scenario we will not use RTS/CTS
  - We only have base station with clients

802.11 DCF example



Transport layer features

- We will use both UDP and TCP transport protocols
  - UDP does not really add anything, one must anyway use a traffic source on top of UDP
- TCP features
  - Implements reliable transport
    - Receiver sends ACKs
    - Bi-directional communication
  - Window based flow/congestion control
    - Window size defines an upper bound on the number of unacknowledged packets that can be in the network
    - Transmission rate ~ window/RTT
  - TCP congestion control principles
    - Idea: modify window size adaptively based on “available capacity”
    - AIMD: window grows linearly until at packet loss it is halved
  - Fairness: TCP fairness results from the principle that packets can be only sent after receiving ACKs (if ACKs stop coming nothing can be sent)
    - Self-clocking mechanism

Tasks with static traffic

- In these tasks the aim is to investigate the efficiency of 802.11 DCF mechanism with different transport protocols
  - Performance depends on packet size and properties of UDP/TCP
- Task 1
  - Simulate a greedy CBR source over 802.11
  - Measure the throughput
    - Requires analysis of the trace file
  - Parameter: packet size
  - Skeleton file: task-1.tcl
- Task 2
  - Use a greedy TCP source over 802.11
  - Measure the throughput
    - Measuring more easy
  - Parameter: packet size
  - Skeleton file: task-2.tcl

Task 3: flow level simulation (1)

- Flow-level model
  - Model for elastic traffic (file transfers as controlled e.g. by TCP)
  - Dynamioc system with randomly arriving flows sharing the resources, the flows have random sizes
  - Performance = mean time to transmit whole file or average throughput
  - Models the performance of data traffic

- Task 3
  - We simulate random TCP flows/file transfers over 802.11
  - Requests for file transfers arrive according to a Poisson process with rate λ
  - Theoretical capacity C = 11 Mbps
  - Mean file size B = 400 packets (exponentially distributed), packet size 1460B (+ 40B of IP overhead)
  - Study mean file transfer delay as a function of load, $\rho = \lambda * (B / C)$

S-38.3148 Simulation of data networks / ns2

Task 3: flow level simulation (2)

- Idealized model for TCP
  - Assume that TCP shares bandwidth perfectly fairly (ok in our case)
  - Rate adaptation is instantaneous
    - with N(t) flows in system, each flow always gets C/N(t)
  - Flows arrive according to a Poisson process

$\Rightarrow$ Processor sharing model

  - The system is stable if $\rho < 1$ (i.e., the mean delay $< \infty$)
  - Compare the simulations with TCP over 802.11 to above idealized system

- Schedule
  - 1st question session: Fri, 30.11., at 14 – 16, in Maari-M (Maarintalo)
  - 2nd question session: Tue, 11.12., at 14 – 16, in Maari-M (Maarintalo)
  - Deadline: Fri, 21.12., at 12:00

Task 3 skeleton

- Flow level simulations of TCP
  - event scheduling handled from Otcl level
  - scheduling concerns arrival and departure of flows
  - a skeleton code for handling this is given
  - the skeleton code is in file task-3.tcl
- Your task is to…
  - create the topology,
  - implement the main program for controlling the simulation,
  - implement the final computation of performance statistics

Some hints for programming (1)

- Creating an array of TCPs
  - you can create an array in TCL without declaring it first
  - example: creating 10 TCPs, configuring them and storing them in the array tcp()

```
for {set nn 0} {$nn < 10} {incr nn} {
  set tcp_s($nn) [new Agent/TCP/Reno]
  $tcp_s($nn) set packetSize_ 1460
  $tcp_s($nn) set window_ 1000
  $tcp_s($nn) set fid_ $nn
  . . .
}
```

  - multidimensional arrays: for example, `$tcp_s(2,3)` = tcp-agent in class 2 and id 3

Some hints for programming (2)

- Accessing lists
  - lists can be initialized easily
  - operations for lists:
    - `llength` : length of the list
    - `lindex` : pick element at given index from the list
    - `lappend` : insert element
    - `lreplace` : search and replace
  - Example:

```
set a {1 2 3 4}
set b [lindex $a 1] (=> b = 2, indexing starts from 0)
lappend $a 5 (=> a = {1 2 3 4 5})
```