An Introduction of Performance Evaluation using ns-2 Simulator

Kien Nguyen

National Institute of Information and Communications Technology, Japan

Aizu, December 2015

Outline

Outline

Network Performance Evaluation

- Network performance refers measures of service quality of a network as seen by the customer.
  - Or key criterion (i.e., performance metrics) in the design and/or use of networked systems
- Performance evaluation is a constructive process to compare a number of alternative designs to find the best design

Performance Evaluation Techniques

Suppose you devise a great protocol. How do you show that it is great?

1. Experiments
   - In a test-bed network or an operational network
   - e.g., put all routers together and let people use them
2. Analytical Modeling
   - Use mathematical notions and models
   - e.g., model routers using graph theory
3. Simulation
   - Use programming to represent routers
   - e.g., C++, ns-2

Why Simulation?

Comparison of techniques

	Analytical Modeling	Simulation	Experiments
Accuracy	Moderate	Moderate	Various
Cost	Small	Medium	High
Time required	Small	Medium	High

Pros and Cons

	Analytical Modeling	Simulation	Experiments
Pros	Insight	Easy(Cheap)	Realistic
		Used for verification	
Cons	Need to make assumption	Not much insight	Expensive
			Sometime not possible

When

- Real-system not available, is complex/costly or dangerous
- Quickly evaluate design alternatives (e.g., different system configurations)
- Evaluate complex functions for which closed form formulas or numerical techniques not available

Simulation Overview

1. Platform: hardware, software, or hybrid
2. Developer: commercial or in-house
3. Source code: open or close
4. Paradigm: Time-dependent/non-time-dependent; Time-driven/event-driven
5. Dimension of simulation performance (Scalability, Fidelity, Execution Speed, etc.)
6. Physical layer: Matlab, Labview, etc.
7. Network layer: ns-2, Opnet, Qualnet

Simulation Overview

Three Simulation Main Steps

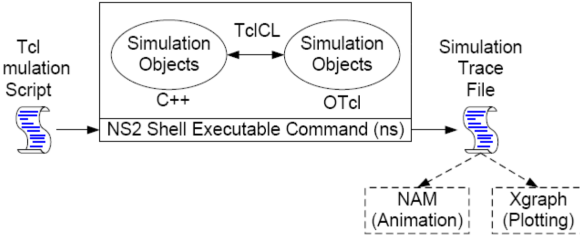
1. Design and Implementation
   - Things to simulate
   - Assumptions
   - Performance measure
   - Code Implementation
2. Simulation
   - Network Configuration Phase
   - Simulation Phase
3. Result Compilation
   - Debugging and Tracing
   - Compute performance measures

Outline

Ns-2

- ns-2 stands for network simulator version 2
- `http://www.isi.edu/nsnam/ns/`
- ns-2
  - is a discrete event simulator for networking research
  - works at packet level
  - provides support to simulate protocols such as TCP, UDP, FTP, HTTP and DSR, etc.
  - simulate wired and wireless network

Ns-2 Architecture



- C++: Internal mechanism
  - Network protocol stack written in C++
- OTcl: User interface
  - Tcl (Tool Command Language) used for specifying scenarios and events
- TclCL: Connecting C++ to OTcl

Installation

1. Access ns-2 website
   - Download link
     `http://www.isi.edu/nsnam/ns/ns-build.html`
2. Get all in one packet (e.g., ns-allinone-2.35.tar.gz)
   - ns2, Tcl/Tk, OTcl, TclCL
   - nam, Zlib, Xgraph
3. Unzip all the files and run the install script "./install"
   - Follow the instruction
   - ns-2 is designed for Unix
   - For windows, Cygwin required

Simple scenario


```
#Create a simulator object
set ns [new Simulator]

#Open trace file
set f [open out.tr w]
$ns trace-all $f

#Create two nodes
set n0 [$ns node]
set n1 [$ns node]

#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

Simple scenario


```
#Create a simulator object
set ns [new Simulator]

#Open trace file
set f [open out.tr w]
$ns trace-all $f

#Create two nodes
set n0 [$ns node]
set n1 [$ns node]

#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

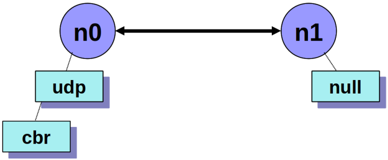
Simple scenario


```
#Create a simulator object
set ns [new Simulator]

#Open trace file
set f [open out.tr w]
$ns trace-all $f

#Create two nodes
set n0 [$ns node]
set n1 [$ns node]

#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

simple.tcl

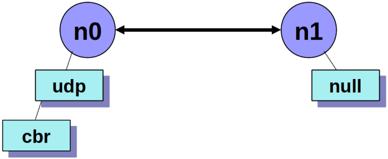
```
#Create a simulator object
set ns [new Simulator]
#Open trace file
set f [open out.tr w]
$ns trace-all $f
#Define a 'finish' procedure
proc finish {} {
       global ns
       $ns flush-trace
       exit 0
}
#Create two nodes
set n0 [$ns node]
set n1 [$ns node]
#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

Adding traffic



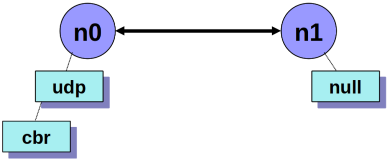

```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

#Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0

#Schedule events for the CBR agent
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

Adding traffic



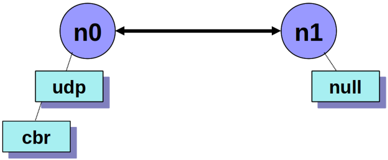

```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

#Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0

#Schedule events for the CBR agent
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

Adding traffic




```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

#Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0

#Schedule events for the CBR agent
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

Adding traffic




```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

#Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0

#Schedule events for the CBR agent
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

Trace analysis

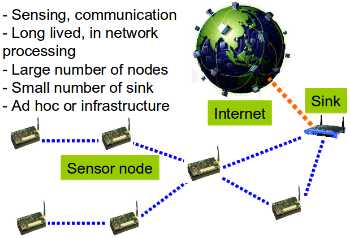
```
...
+ 0.110419 1 2 tcp 1040 ------- 2 1.0 4.0 5 12
+ 0.110419 1 2 tcp 1040 ------- 2 1.0 4.0 6 13
- 0.110431 1 2 tcp 1040 ------- 2 1.0 4.0 5 12
- 0.110514 1 2 tcp 1040 ------- 2 1.0 4.0 6 13
r 0.11308 0 2 cbr 1000 ------- 1 0.0 3.0 2 8
+ 0.11308 2 3 cbr 1000 ------- 1 0.0 3.0 2 8
- 0.11308 2 3 cbr 1000 ------- 1 0.0 3.0 2 8
r 0.11316 0 2 cbr 1000 ------- 1 0.0 3.0 3 9
+ 0.11316 2 3 cbr 1000 ------- 1 0.0 3.0 3 9
- 0.113228 2 3 cbr 1000 ------- 1 0.0 3.0 3 9
```

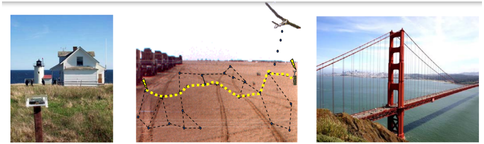
Type Identifier	Time	Source Node	Destination Node	Packet Name	Packet Size	Flags	Flow ID	Source Address	Destination Address	Sequence Number	Packet Unique ID

{enque(+),deque(-),receive(r),drop(d)}

Outline

Wireless Sensor Network

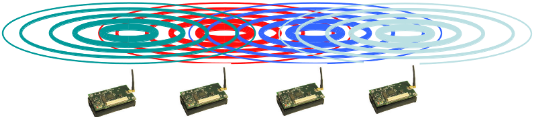




Habitat Monitoring   Object Tracking   Structural Health

- Form Networks of Wireless Sensors (WSNs)
- Limited resources: memory, energy, bandwidth
- Long lifetime requirements: months to tens of years
- Must meet the applications' QoS

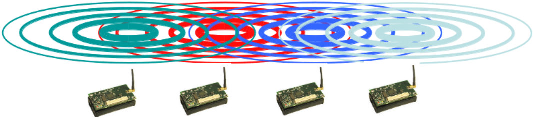
MAC protocol



- The role of medium access control (MAC)
  - Controls when and how each node can transmit in the wireless channel
- Why do we need efficient MACs?
  - Wireless channel is a shared medium
  - Sensor node has limited capabilities

Energy efficient MAC protocols, which guarantee QoS parameters such as latency, throughput, etc., are vital

MAC protocol

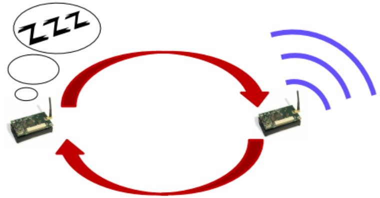


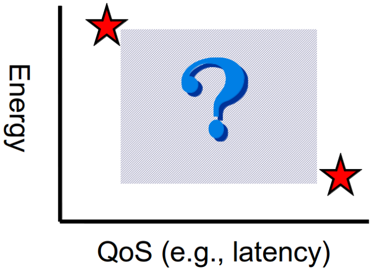
- The role of medium access control (MAC)
  - Controls when and how each node can transmit in the wireless channel
- Why do we need efficient MACs?
  - Wireless channel is a shared medium
  - Sensor node has limited capabilities

Energy efficient MAC protocols, which guarantee QoS parameters such as latency, throughput, etc., are vital

Common on Power Saving MAC Protocol

Duty cycle: the percent of time a node is active



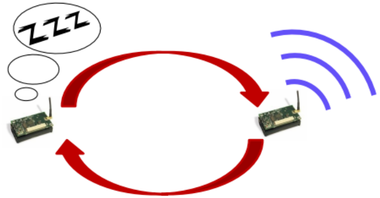


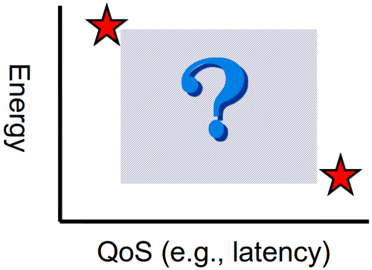
Classification of duty cycling MAC

- Synchronous protocols:
  - Single-hop protocols (e.g., S-MAC, D-MAC)
  - Multi-hop protocols (e.g., R-MAC, AM-MAC, DW-MAC)
- Asynchronous protocols:
  - Sender-initiated protocols (e.g., B-MAC, X-MAC, C-MAC)
  - Receiver-initiated protocols (e.g., RI-MAC, PW-MAC)

Common on Power Saving MAC Protocol

Duty cycle: the percent of time a node is active

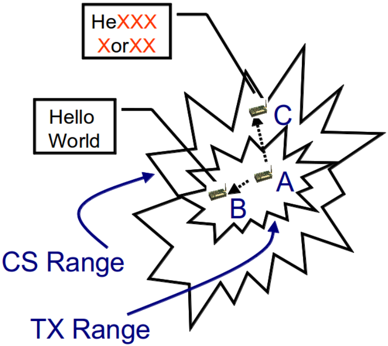




Classification of duty cycling MAC

- Synchronous protocols:
  - Single-hop protocols (e.g., S-MAC, D-MAC)
  - Multi-hop protocols (e.g., R-MAC, AM-MAC, DW-MAC)
- Asynchronous protocols:
  - Sender-initiated protocols (e.g., B-MAC, X-MAC, C-MAC)
  - Receiver-initiated protocols (e.g., RI-MAC, PW-MAC)

Background



Tx_Range: Transmission Range; CS_Range: Carrier Sensing Range

- Broadcast nature: a node can “overhear” the content of a packet transmitting within its transmission range
- Carrier sensing: a node can sense the channel busy when there exist a data transmission within its carrier sensing range
- “Listen before talk” to avoide collision (i.e., use control packets request-to-send (RTS), clear-to-send (CTS))

Design MAC^2

Objectives

- High energy efficiency
- Low latency
- High throughput
- Effectively delivery packets
- Efficient under a wide range of traffic loads
- Example of application: sensing event centric

MAC^2

MAC^2: Multi-hop Adaptive MAC with Packet Concatenation[a]

[a] **Kien Nguyen**, Ulrich Meis, and Yusheng Ji, *"MAC^2: A Multi-hop Adaptive MAC Protocol with Packet Concatenation for Sensor Networks,"* IEICE Trans. on Sys. & Info., No.2, pp. 480-489, Feb. 2012

Design MAC^2

Objectives

- High energy efficiency
- Low latency
- High throughput
- Effectively delivery packets
- Efficient under a wide range of traffic loads
- Example of application: sensing event centric

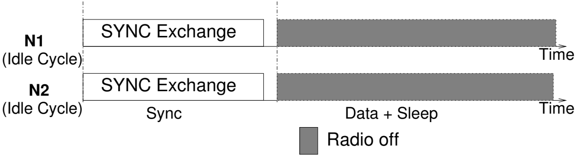
MAC^2

MAC^2: Multi-hop Adaptive MAC with Packet Concatenation[a]

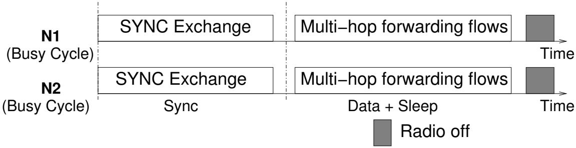
[a] **Kien Nguyen**, Ulrich Meis, and Yusheng Ji, *"MAC^2: A Multi-hop Adaptive MAC Protocol with Packet Concatenation for Sensor Networks,"* IEICE Trans. on Sys. & Info., No.2, pp. 480-489, Feb. 2012

Adaptive Scheme in Sync Period

- The schemes uses the first bit of SYNC packet as signal (0 = Normal Sync, 1 = Signaling Sync)
- When there is no data, nodes follow an idle cycle



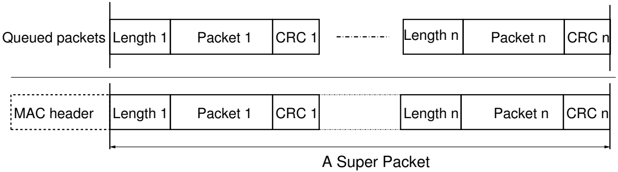
- When there is a data packet, nodes follow a busy cycle
- Assumption: Signaling SYNCs always win the channel



Packet Concatenation

Why?

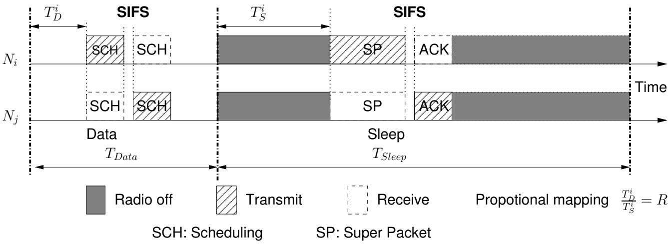
- Packet are always queued
- Packet size is small
- All packets are usually addressed to the same sink
  - The major requirement of packet concatenation in our work



How?

- Several packets that share the destination field are concatenated into a super packet
- The length of a super packet (l_{SP}) is always less than or equal to the concatenation threshold l_{TH}

Data Transmission in a Busy Cycle



Demand Wakeup Transmission

- The manner is original from Demand Wakeup MAC (DW-MAC) ($R_{org} = \frac{T_{Sleep}}{T_{Data}}$)
- MAC^2 utilizes packet concatenation scheme
- N_i and N_j successfully exchange SCHs. T_D^i will be used to schedule the time wakeup (T_S^i) in the upcoming Sleep period:

$$\frac{T_S^i}{T_D^i} = \frac{l_{ACK} + l_{TH} + l_{SIFS}}{l_{SCH} + l_{SIFS}} = R_{min}$$

(l_{SP}, l_{ACK}, l_{SCH}, l_{SIFS}: length of Super Packet, ACK, SCH and SIFS)

Numerical Analysis

Using R_{min}

1. In the Sleep period there will be no collision at the intended receivers with all size of SP if $R \geq R_{min}$
2. MAC^2 protocol achieves minimum latency at R_{min}

Bounds of latency in low traffic environments

1. Lower bound
2. Upper bound in single source scenario
3. Upper bound in M sources scenario

MAC^2 implementation

Modify/create the related files

- e.g., mac2.cc, mac2.h (MAC layer)
- wireless-phy.cc, wireless-phy.h (PHY layer)
- Expected trace information (correctly and efficiently collect performance information)
- Add new object to Makefile
- Compile/Re-compile
- Editor/debugger (command line or Eclipse)

Coding/Debugging takes much time

MAC^2 implementation

Modify/create the related files

- e.g., mac2.cc, mac2.h (MAC layer)
- wireless-phy.cc, wireless-phy.h (PHY layer)
- Expected trace information (correctly and efficiently collect performance information)
- Add new object to Makefile
- Compile/Re-compile
- Editor/debugger (command line or Eclipse)

Coding/Debugging takes much time

Simulation parameter (ns2parameters.tcl)

Network parameters

Rx Power	22.2 mW	Sleep Power	3 μW
State Transition Power	31.2 mW	Tx Power	31.2 mW
Idle Power	22.2 mW	Transmission Range	250 m
Carrier Sensing Range	550 m	l_{SIFS}	5 ms
Contention Window (CW)	64 ms	l_{DIFS}	10 ms
Retry Limit	5	l_{DATA}	43 ms
l_{ACK}	11 ms	l_{SCH}	14.2 ms
l_{TH}	243 ms	ifq	2500 bytes

Time duration parameters

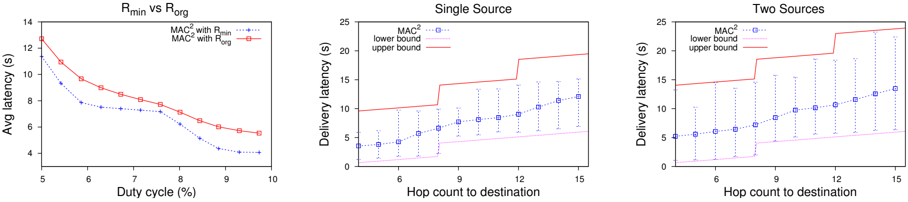
	T_{Cycle}	T_{Sync}	T_{Data}	T_{Sleep}
DW-MAC/MAC^2	4465 ms	55.2 ms	168 ms	4241.8 ms

Duty cycle

$DC = (T_{Sync} + T_{Data})/T_{Cycle}$ is 5 % in grid and random network scenarios

Latency Bound (verifying numerical analysis)

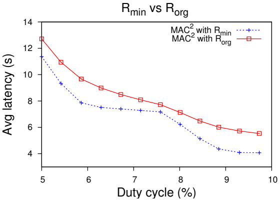
- Chain scenario (200 meters between two neighboring nodes is 200 meters) (chain_multihop.tcl)
- The sources send 100 packets to the farthest node at the packet rate one packet/30 seconds (gen_traffic_sink_statistics.tcl)

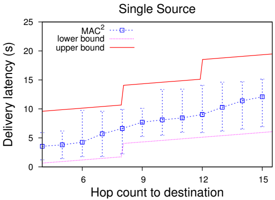


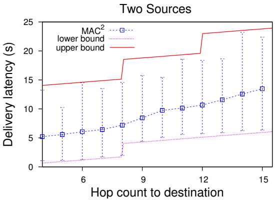
The simulation results confirm our analysis

Latency Bound (verifying numerical analysis)

- Chain scenario (200 meters between two neighboring nodes is 200 meters) (chain_multihop.tcl)
- The sources send 100 packets to the farthest node at the packet rate one packet/30 seconds (gen_traffic_sink_statistics.tcl)



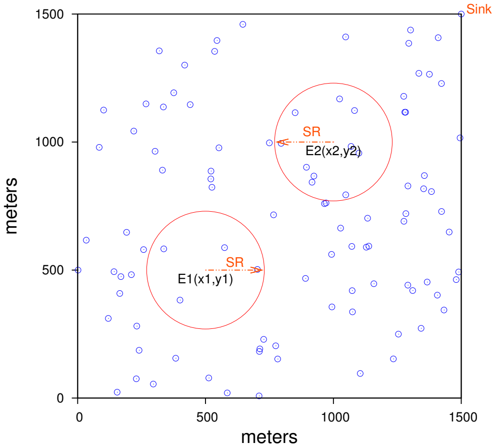




The simulation results confirm our analysis

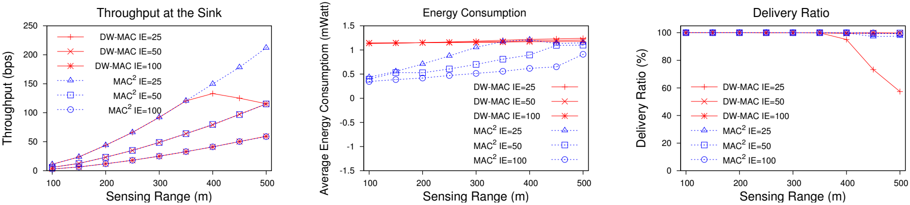
Random Correlated Event (RCE) Traffic Model

- An event is generated at a random location (x, y) (e.g., tra_49square200_interval_30_event100_eventradius_500_pktsize28.tcl)
- A node sends one packet if its location is within the circle centered at (x, y) with radius *SR* (Sensing Range)
- The variation of *SR* varies the traffic load



Performance in Grid Scenario

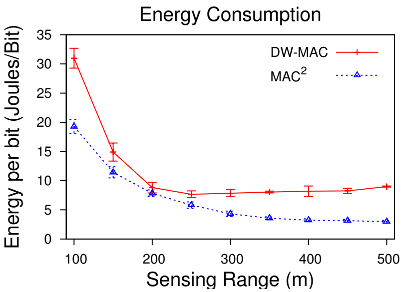
- Total 200 events are generated with different values of the inter-event interval (IE) in a grid scenario of (7×7)
- Each node is 200 meters apart from its direct neighbors, the sink is at the center
- Each value is averaged over ten runs, the error bars show the 95% confidence interval

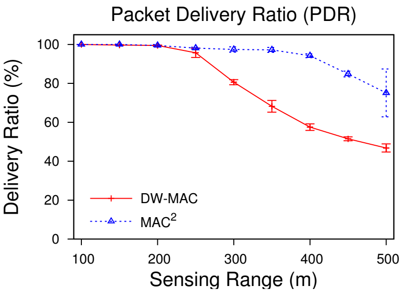


- MAC^2 achieves better delivery ratio than DW-MAC when the traffic is high
- MAC^2's nodes consume less energy than DW-MAC's

Performance in Network Scenario: Energy and PDR

- 100 nodes are randomly deployed in the square 1500 $\times$ 1500 meter; the sink node is at the right top corner
- RCE with 200 events; the inter-event interval is randomly between 10 to 30 seconds
- Each value is averaged over 10 runs; the error bars show the 95% confidence interval

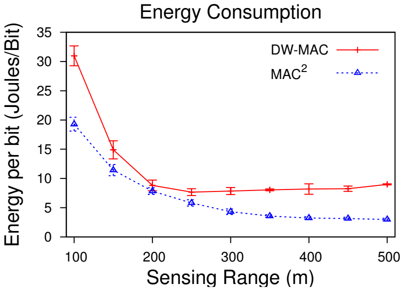


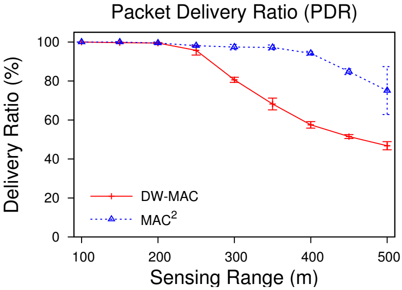


MAC^2 outperforms DW-MAC in all investigated scenarios

Performance in Network Scenario: Energy and PDR

- 100 nodes are randomly deployed in the square 1500 $\times$ 1500 meter; the sink node is at the right top corner
- RCE with 200 events; the inter-event interval is randomly between 10 to 30 seconds
- Each value is averaged over 10 runs; the error bars show the 95% confidence interval

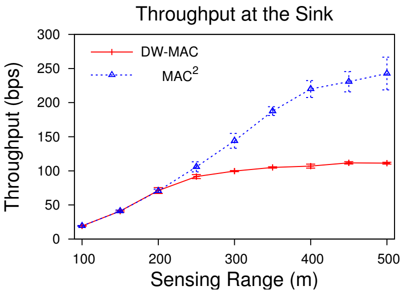


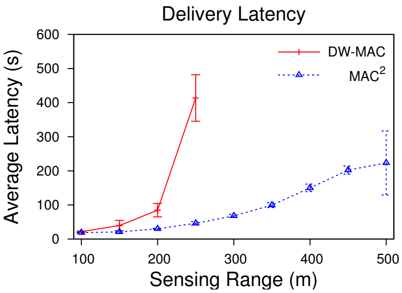


MAC^2 outperforms DW-MAC in all investigated scenarios

Performance in Network Scenario: Throughput and Latency

- When SR is larger than 300 meters, the value of DW-MAC's latency is extra large and is not shown

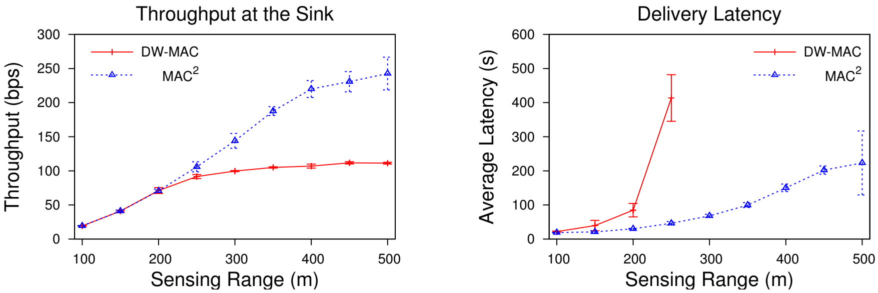




The packet concatenation scheme lets the network in a steady state

Performance in Network Scenario: Throughput and Latency

- When SR is larger than 300 meters, the value of DW-MAC's latency is extra large and is not shown



The packet concatenation scheme lets the network in a steady state

AQ-MAC Design

Objectives

- High energy efficiency
- Efficient under a wide range of traffic loads
- Quality of Service (QoS) provisioning
  - A sensor board contains several types of sensors, which may generate different types of data
  - The data needs an effective transmission strategy

AQ-MAC

The protocol is a combination between a receiver-initiated transmission and an adaptive sleeping method

AQ-MAC Design

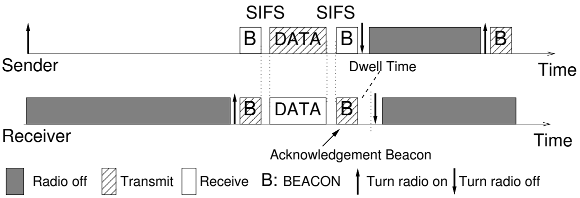
Objectives

- High energy efficiency
- Efficient under a wide range of traffic loads
- Quality of Service (QoS) provisioning
  - A sensor board contains several types of sensors, which may generate different types of data
  - The data needs an effective transmission strategy

AQ-MAC

The protocol is a combination between a receiver-initiated transmission and an adaptive sleeping method

AQ-MAC: Asynchronous QoS-aware MAC [1]



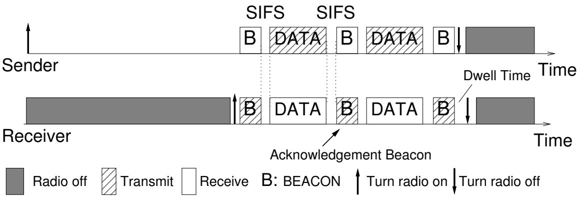
Transmission completed after receive an ack beacon

Receiver-Initiation + QoS Awareness + Packet Concatenation

- The transmission manner is original from Receiver-Initiated MAC (RI-MAC)
  - Node with pending data listens to the channel waits for an incoming packet
  - The sender wastes energy during the waiting time
  - The transmission is suitable with time-critical (high priority) traffic
- The low priority (LP) packets are kept in a queue until a high priority (HP) arrives or after a time out value T_h, ($T_h = \frac{T}{h}$, h: number of hops to the sink)
- The queued packets are concatenated before sending out

[1] **Kien Nguyen** and Yusheng Ji, *“Asynchronous MAC Protocol with QoS Awareness in Wireless Sensor Networks,”* In Proceeding of IEEE GLOBECOM 2012, Dec. 2012

AQ-MAC: Asynchronous QoS-aware MAC [1]



Ack beacon invites new incoming data

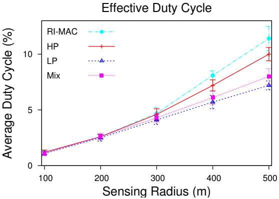
Receiver-Initiation + QoS Awareness + Packet Concatenation

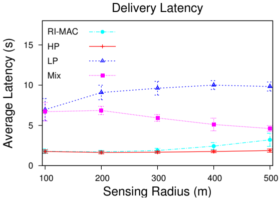
- The transmission manner is original from Receiver-Initiated MAC (RI-MAC)
  - Node with pending data listens to the channel waits for an incoming packet
  - The sender wastes energy during the waiting time
  - The transmission is suitable with time-critical (high priority) traffic
- The low priority (LP) packets are kept in a queue until a high priority (HP) arrives or after a time out value T_h, ($T_h = \frac{T}{h}$, h: number of hops to the sink)
- The queued packets are concatenated before sending out

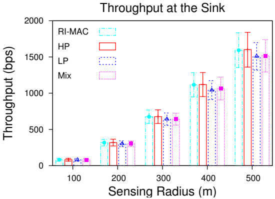
[1] **Kien Nguyen** and Yusheng Ji, *"Asynchronous MAC Protocol with QoS Awareness in Wireless Sensor Networks,"* In Proceeding of IEEE GLOBECOM 2012, Dec. 2012

AQ-MAC Evaluation

- In 7 $\times$ 7 grid scenario; a sink at the center; 100 events; inter-events (0, 5) seconds
- In the case of Mix traffic, 50 events generate HP packets and 50 others generate LP



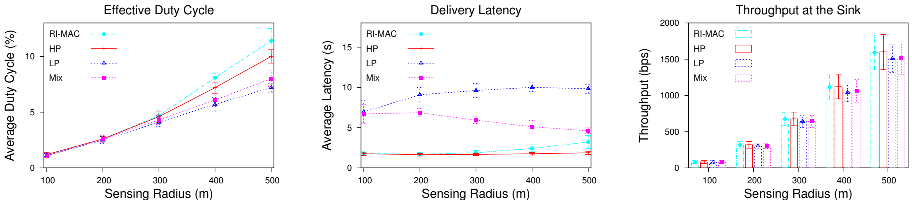




1. AQ-MAC with HP achieves better than RI-MAC in terms of energy efficiency
2. AQ-MAC adapts well with the variation of traffic in the multi-hop network

AQ-MAC Evaluation

- In 7 $\times$ 7 grid scenario; a sink at the center; 100 events; inter-events (0, 5) seconds
- In the case of Mix traffic, 50 events generate HP packets and 50 others generate LP



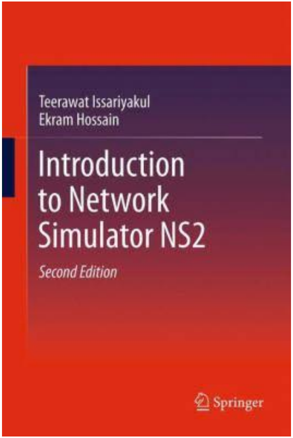
1. AQ-MAC with HP achieves better than RI-MAC in terms of energy efficiency
2. AQ-MAC adapts well with the variation of traffic in the multi-hop network

Outline

Conclusion

- A brief introduction about network performance evaluation
  - Emphasizing on the simulation-based approach
- An overview of the network simulator ns-2
  - Basic information
  - A simple example
- Using ns-2 in a real research work
  - Simulating MAC protocols in wireless sensor networks

References



- ns-2 book by Teerawat Issariyakul
  - And related websites (blog, homepages)
- Network Performance and NS2 by Mohammed M. Kadhum
  - Presented at NETAPPS2010

Thank You & Questions?

Kien Nguyen
kienng@nict.go.jp