A Centralized Token-based Medium Access Control Mechanism for Wireless Network-on-Chip

EL7044 - Advanced Wireless Networking Concepts
Alberto Castro R.
August 22, 2024

Outline

- Introduction
- Background
- Centralized Token-based MAC Mechanism
- Implementation and Results
- Conclusion
- Simulator
- Demo
    - Noxim, NoC simulator
    - Omnet++

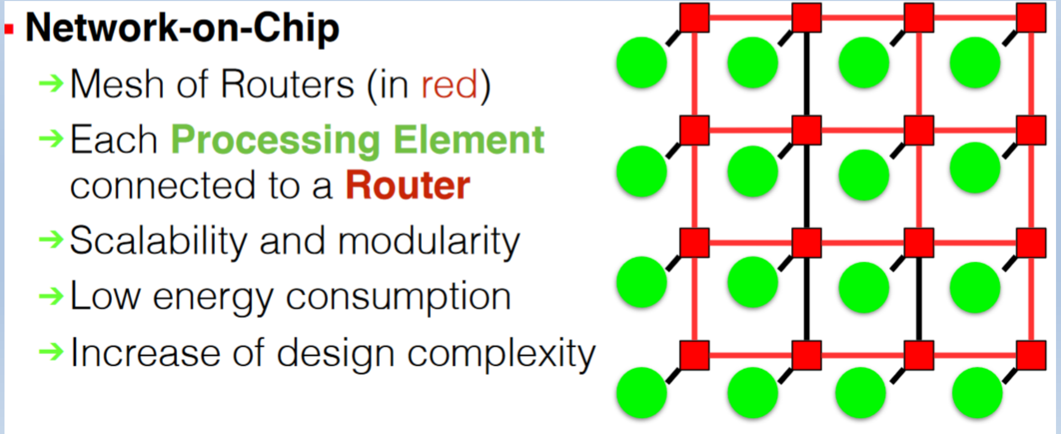
Introduction

- Overview of Centralized Token-based Medium Access Control
- Token-passing in Wireless Network-on-Chip (WiNoC)
- Keywords and Definitions: C-MAC, Network-on-Chip, Wireless interconnect
- Objectives and Contributions of the Proposed C-MAC mechanism

Background

- Challenges in Medium Access Control (MAC) in WiNoC
- Comparison of Different MAC Mechanisms (CSMA, CDMA, Token-based)
- Importance of Fair Scheduling and Bandwidth Utilization in WiNoC
- Previous Works in Dynamic MAC Mechanisms for WiNoC

Paper: **Wireless NoC as Interconnection Backbone for Multicore Chips Promises and Challenges**



Centralized Token-based MAC Mechanism

- Design and Operation of C-MAC
- Advantages of Prioritizing RHs with Most Packets
- Simulation Results and Performance Evaluation of C-MAC
- Comparison with Existing MAC Mechanisms: CM –conventional method-; RACM –radio access control mechanism-.

Algorithm 1: Proposed C-MAC Mechanism

Input: RH_T, RH_i, RH_R, packet, is_token_RH

Output: Token

```
CMAC = {}
While (is_ token_RH)
    for ∀ RHi ∈ RH_T
        RH_R = RH_T - CMAC
        for ∀ RHi ∈ RH_R
            Choose RH_i with most packet
            Assign Token to RH_i
        end for
        Add RH_i to CMAC
        if RH_R = 1
            CMAC = {}
        end if
    end for
end while
```

Implementation and Results

- Description of WiMesh Architecture with C-MAC Module

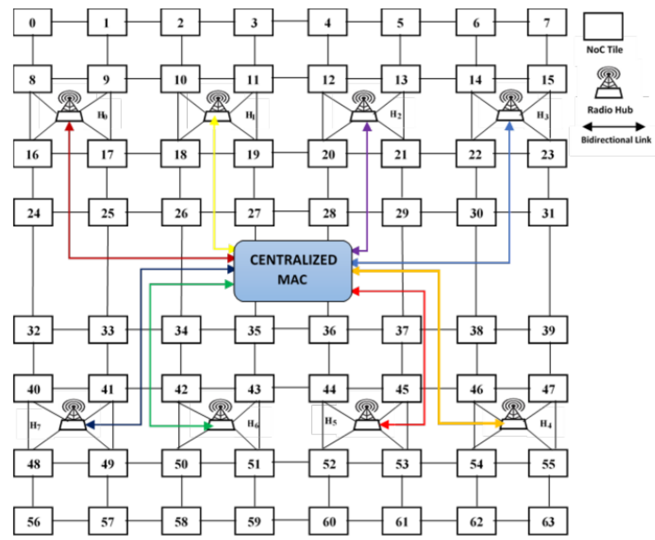
- Interface and Operation of C-MAC Module

- Simulation Setup and Parameters

- Analysis of Results and Discussion on Network Throughput and Latency

Table I: SIMULATION SETUP

PARAMETER	DESCRIPTION
Network Sizes	8*8 (64 cores)
Number of Radio Hub	4*4
Synthetic Traffic Distribution	Hotspot, Random, Shuffle, and Transpose
Application Traffic Distribution	Fluidanimate, Blackscholes, Freqmine, and Swaption
Number of Channels	1
Simulation_Time Cycles	100 000
Technology	65 nm
Clock Frequency	1 Ghz
Switching Mechanism	Wormhole
Radio Access Control	Conventional Token Ring, RACM, Propose C-MAC
Selection Strategies	Random
Flit Size	32 bits
Routing Algorithm	XY
Wireless Data Rate	16Gbps
Wireless Communication	Millimeter-Wave



Results

The proposed C-MAC shows better performance on the synthetic traffic patterns, at an average of 37% throughput improvement than the conventional method and 11% compared to RACM, while it has average network latency at 12.75% compared to the conventional method and 6.25% compared to RACM

Conclusion

Traffic Patterns	PIR	CM	RACM	CMAC
Hotspot	$1 * 10^{-5}$	0.005	0.007	0.007
	$1 * 10^{-4}$	0.051	0.063	0.069
	$1 * 10^{-3}$	0.511	0.634	0.683
	$2 * 10^{-3}$	0.628	0.779	0.842
Random	$1 * 10^{-5}$	0.005	0.006	0.007
	$1 * 10^{-4}$	0.050	0.063	0.068
	$1 * 10^{-3}$	0.509	0.633	0.686
	$2 * 10^{-3}$	0.930	1.150	1.238
Shuffle	$1 * 10^{-5}$	0.005	0.007	0.007
	$1 * 10^{-4}$	0.051	0.063	0.068
	$1 * 10^{-3}$	0.515	0.635	0.681
	$2 * 10^{-3}$	0.941	1.165	1.256
Transpose	$1 * 10^{-5}$	0.005	0.006	0.007
	$1 * 10^{-4}$	0.051	0.064	0.068
	$1 * 10^{-3}$	0.514	0.633	0.684
	$2 * 10^{-3}$	1.020	1.268	1.360

Summary of Findings and Contributions

- C-MAC mechanism improves the utilization of the radio medium in WiNoCs, by dynamically adjusting the token of RHs allocation based on the most packets (<- "rating") in each RHs in the cycle.

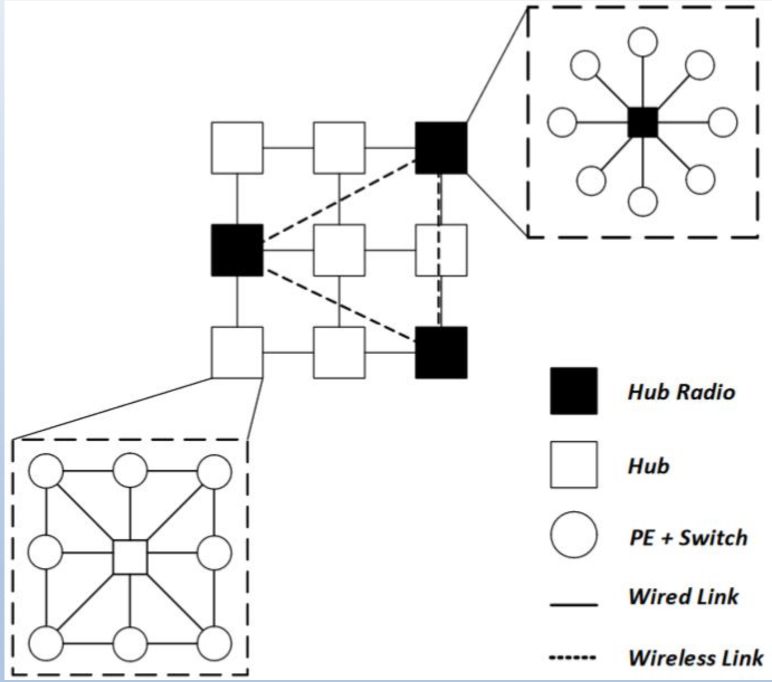
- The simulation results under synthetic and application traffic patterns show that the proposed C-MAC arbitration mechanism performs better than the conventional method and RACM by reducing.

- (Note: Below are the tested simulator alternatives, in addition to Noxim, the WinNoC simulator used in the paper.)

Noxim, WiNoC simulator

- Cycle accurate, Open source
- SystemC signals level simulation in C++
- Performance & Power estimation
- Modular plugin-like addition of Routing/Selection strategies
- Wireless transmissions simulation

https://github.com/davidepatti/noxim

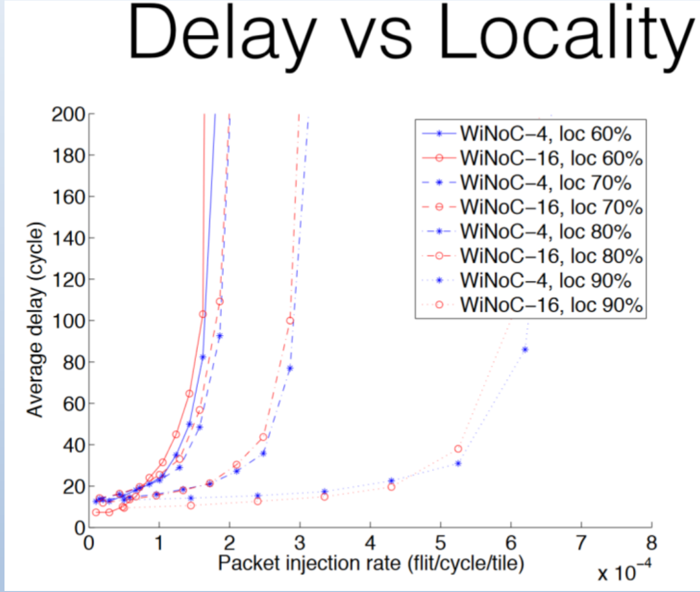


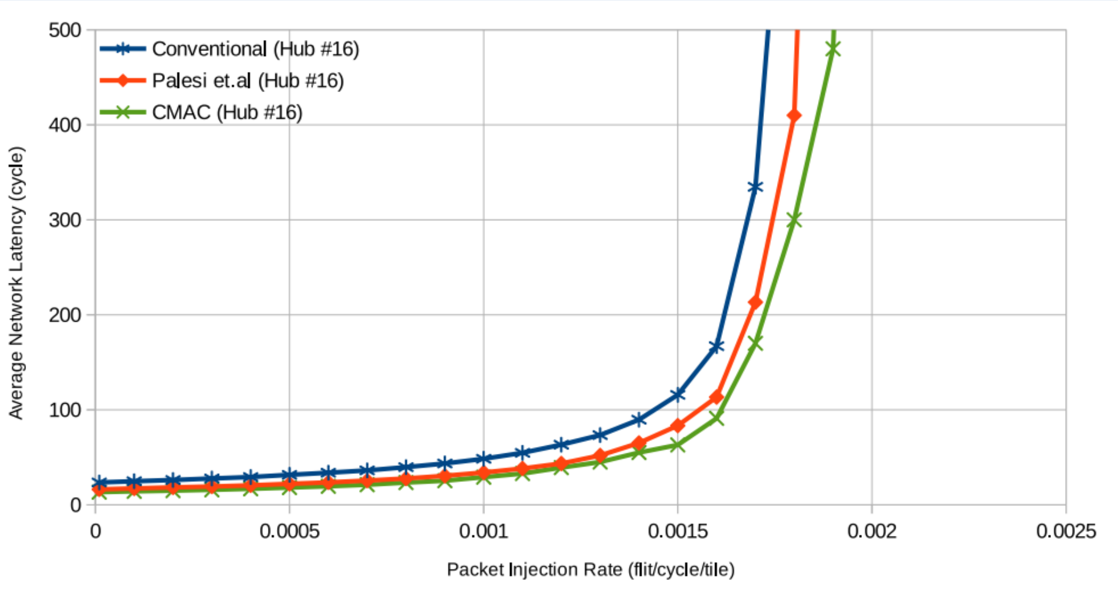
Noxim, WiNoC simulator - demo

```
# WIRELESS CONFIGURATION
#
Hubs:
    defaults:
    # channels from which Hub can receive/transmit
        rx_radio_channels: [0]
        tx_radio_channels: [0]
    # list of node tiles attached to the hub
        attached_nodes: []
    # size of buffers connecting the hub to tiles
        to_tile_buffer_size: 4
        from_tile_buffer_size: 4
    # size of antenna tx/rx
        rx_buffer_size: 64
        tx_buffer_size: 64

# for each hub, the same parameters specified above can be customized
# If not specified, the above default values will be used
# What is usually needed to be customized specifically for each hub is
# the set of nodes that are connected to it. In this simple topology
# we have 4 hubs (0-3) connected to the four nodes of the 2x2
# sub-meshes

    0:
        attached_nodes: [49,50,65,66]

    1:
        attached_nodes: [53,54,69,70]
```

```
root@Alberto-Castro-R:~/noxim/bin# ./noxim -config ../config_examples/256_8h.yaml

        SystemC 2.3.1-Accellera --- Aug 21 2024 22:21:08
        Copyright (c) 1996-2014 by all Contributors,
        ALL RIGHTS RESERVED
        --------------------------------------------
                Noxim - the NoC Simulator
                (C) University of Catania
        --------------------------------------------
Catania V., Mineo A., Monteleone S., Palesi M., and Patti D. (2016) Cycle-Accurate Network on Chip Simulation with Noxim
. ACM Trans. Model. Comput. Simul. 27, 1, Article 4 (August 2016), 25 pages. DOI: https://doi.org/10.1145/2953878


Loading configuration from file "../config_examples/256_8h.yaml"... Done
Loading power configurations from file "power.yaml"... Done
Reset for 1000 cycles...  done!
 Now running for 10000 cycles...
Noxim simulation completed. (11000 cycles executed)

% Total received packets: 13458
% Total received flits: 161546
% Received/Ideal flits Ratio: 0.584295
% Average wireless utilization: 0
% Global average delay (cycles): 1773.34
% Max delay (cycles): 8845
% Network throughput (flits/cycle): 17.9496
% Average IP throughput (flits/cycle/IP): 0.0701155
% Total energy (J): 3.82392e-05
%       Dynamic energy (J): 6.63516e-06
%       Static energy (J): 3.1604e-05
```

Noxim, WiNoC simulator



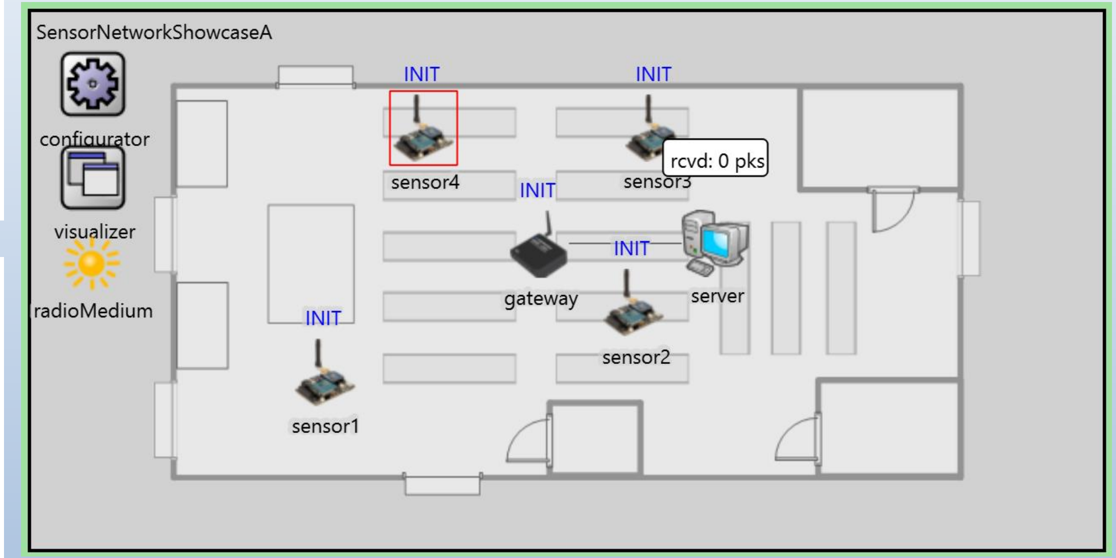


Average Latency of the 8*8-core with 16_RHs under Synthetic traffic patterns

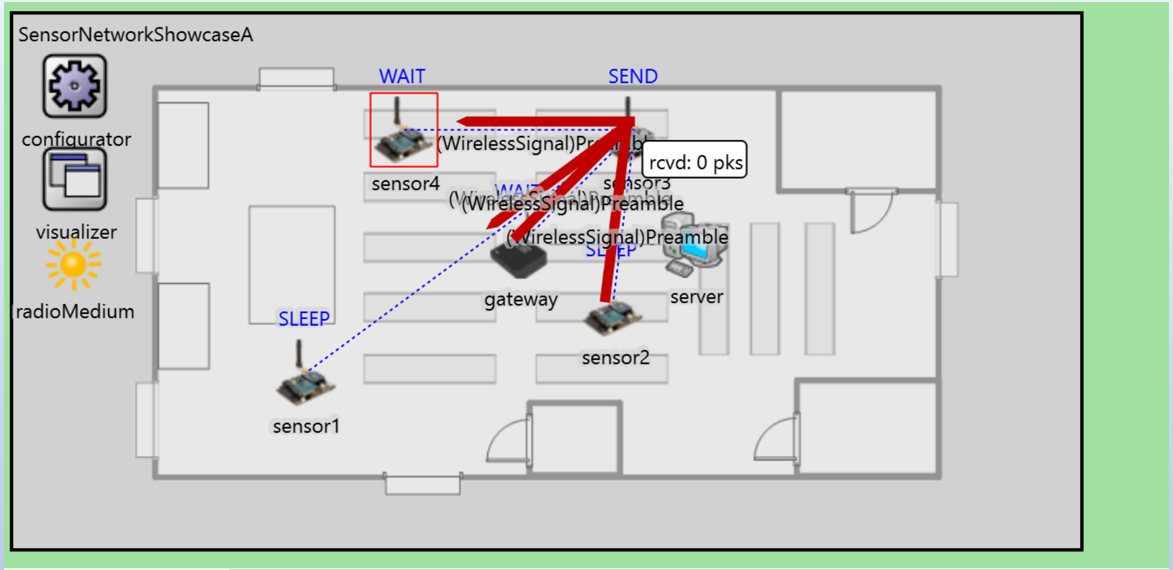
Another alternative, Omnet++

```
[Config BMac]
network = SensorNetworkShowcaseA

**.display-name =
**.wlan[*].mac.typename = "BMac"
**.wlan[*].mac.headerLength = 1B
**.wlan[*].mac.slotDuration = 0.025s
**.wlan[*].queue.typename = "DropTailQueue"
**.wlan[*].queue.packetCapacity = 20
```

```
# radio and radioMedium
**.radio.centerFrequency = 2.45GHz
**.radio.bandwidth = 2.8MHz

**.radio.transmitter.bitrate = 19200 bps
**.radio.transmitter.headerLength = 8b
**.radio.transmitter.preambleDuration = 0.0001s
**.radio.transmitter.power = 2.24mW

**.radio.receiver.energyDetection = -90dBm
**.radio.receiver.sensitivity = -100dBm
**.radio.receiver.snirThreshold = -8dB

*.radioMedium.backgroundNoise.power = -110dBm

**.wlan[*].mac.headerLength = 8b
```



Execute Omnet++, wireless sensor networks showcase



Event#	Time	Relevant Hops	Name	TxUpdate? / Sour	Length / Destinat	Info / Protocol	Type
#45	0.107'663'148'995	sensor3 ⟶ sensor4	Preamble	0A-AA-00-00-00-04	FF-FF-FF-FF-FF		
#45	0.107'663'148'995	sensor3 ⟶ sensor1	Preamble	0A-AA-00-00-00-04	FF-FF-FF-FF-FF		
#45	0.107'663'148'995	sensor3 ⟶ sensor2	Preamble	0A-AA-00-00-00-04	FF-FF-FF-FF-FF-FF	B-MAC	
#45	0.107'663'148'995	sensor3 ⟶ gateway	Preamble	0A-AA-00-00-00-04	FF-FF-FF-FF-FF-FF	B-MAC	
#56	0.112'663'148'995	sensor3 ⟶ sensor4	Preamble	0A-AA-00-00-00-04	FF-FF-FF-FF-FF-FF	B-MAC	
#56	0.112'663'148'995	sensor3 ⟶ sensor1	Preamble	0A-AA-00-00-00-04	FF-FF-FF-FF-FF-FF	B-MAC	
#56	0.112'663'148'995	sensor3 ⟶ sensor2	Preamble	0A-AA-00-00-00-04	FF-FF-FF-FF-FF-FF	B-MAC	
#56	0.112'663'148'995	sensor3 ⟶ gateway	Preamble	0A-AA-00-00-00-04	FF-FF-FF-FF-FF-FF	B-MAC	
#69	0.117'663'148'995	sensor3 ⟶ sensor4	Preamble	0A-AA-00-00-00-04	FF-FF-FF-FF-FF-FF	B-MAC	
#69	0.117'663'148'995	sensor3 ⟶ sensor1	Preamble	0A-AA-00-00-00-04	FF-FF-FF-FF-FF-FF	B-MAC	