Fuzzy-Token: An Adaptive MAC Protocol for Wireless-Enabled Manycores

Antonio Franques*, Sergi Abadal†, Haitham Hassanieh*, and Josep Torrellas*
*University of Illinois at Urbana-Champaign †Universitat Politècnica de Catalunya
franque2@illinois.edu, abadal@ac.upc.edu, {haitham, torrella}@illinois.edu

***Abstract*—Recent computer architecture trends herald the arrival of manycores with over one hundred cores on a single chip. In this context, traditional on-chip networks do not scale well in latency or energy consumption, leading to bottlenecks in the execution. The Wireless Network-on-Chip (WNoC) paradigm holds considerable promise for the implementation of on-chip networks that will enable such highly-parallel manycores. However, one of the main challenges in WNoCs is the design of mechanisms that provide fast and efficient access to the wireless channel, while adapting to the changing traffic patterns within and across applications. Existing approaches are either slow or complicated, and do not provide the required adaptivity.**

In this paper, we propose FUZZY TOKEN, a simple WNoC protocol that leverages the unique properties of the on-chip scenario to deliver efficient and low-latency access to the wireless channel irrespective of the application characteristics. We substantiate our claim via simulations with a synthetic traffic suite and with real application traces. FUZZY TOKEN consistently provides one of the lowest packet latencies among the evaluated WNoC MAC protocols. On average, the packet latency in FUZZY TOKEN is 4.4× and 2.6× lower than in a state-of-the art contention-based WNoC MAC protocol and in a token-passing protocol, respectively.

***Index Terms*—Manycore, Wireless NoC, MAC protocol**

I. INTRODUCTION

In recent years, processor manufacturers have steadily increased the number of cores integrated on a processor chip. Currently, Ampere's Altra processor [2] integrates up to 80 ARM cores, AMD's EPYC 7742 processor [1] supports up to 64 cores, and Intel's Xeon Platinum 9282 processor [3] hosts up to 56 cores. It is conceivable that the core count will continue to increase past one hundred.

To serve inter-core traffic, processor chips use Networks-on-Chip (NoCs) [9, 15]. Typically, manycore NoCs are packet-switched networks composed of routers and links arranged in a 2D grid. While this design is more scalable and efficient than buses, the latency and energy consumption of NoCs start to become a problem at these core counts [21, 22]. In particular, messages that need to be broadcasted or traverse a high number of routers before arriving to the intended destination are problematic [14]. The high latency may stall cores as they wait for a response or to synchronize, which throttles execution. Slowdowns of 2–3× have been estimated in processors with several tens of cores [14, 27] and worse effects are expected at higher core counts.

To improve the scalability of NoCs to high core counts, emerging interconnect technologies such as 3D [10], nanophotonics [7], and wireless [6, 28] have been proposed. Among them, wireless technology has shown promise due to its inherent broadcast nature and very low latency for chip-wide transmissions, which is lower than that of conventional NoCs by an order of magnitude [6]. Moreover, no wiring infrastructure is needed, which provides a flexibility that is unattainable with other technologies [13].

The wireless approach is enabled by recent advances in CMOS RF technology, which have allowed the integration of millimeter-wave antennas and transceivers on chip [8, 28]. Based on these, multiple Wireless Network-on-Chip (WNoC) designs have been proposed. Using simulations, these designs are shown to achieve substantial network-level improvements with respect to conventional NoCs [6, 16, 20], and help manycores attain substantial execution speedups and reduced energy consumption [4, 13]. These estimations have assumed wireless bandwidths of a few tens of GHz and low-order modulations, which are both feasible with current technology.

To fully realize the potential of WNoCs, however, it is necessary to build Medium Access Control (MAC) mechanisms that are able to cope with the heterogeneity and performance requirements of the multiprocessor scenario [5]. MAC protocols need to adapt to wide and fast changes in the on-chip traffic, while incurring little to no delay in the transmission. Unfortunately, existing efforts [4, 13, 16, 18, 19] are unable to capture such fast variations, resulting in execution time slowdowns and energy waste.

In this paper, we present FUZZY TOKEN, a new MAC protocol capable of dynamically adapting to the traffic demands of the application, minimizing transmission latency and increasing network throughput. The FUZZY TOKEN algorithm combines the strengths of token passing and contention-based MAC protocols [24] and, with a few simple rules, adapts to different types of workloads. The algorithm is evaluated using both a synthetic traffic model [23], and a set of real application traces. Our simulation-based evaluation shows that FUZZY TOKEN consistently provides one of the lowest packet latencies of the evaluated WNoC MAC protocols. On average, the packet latency in FUZZY TOKEN is 4.4× and 2.6× lower than in a state-of-the art contention-based WNoC MAC protocol and in a token-passing protocol, respectively.

II. BACKGROUND

Traditionally, wired NoC architectures use a packet-switched network with each processor connected to a router as shown in Fig. 1. Each router has several pipeline stages for enqueuing, computing the route, arbitration, and dequeuing packets [22]. Data packets move from source to destination via multiple hops, with each hop incurring delays and energy consumption. A mesh is a typical topology for manycores due to its simple layout and low inter-router link length [22]. However, the average hop count scales with $\sqrt{N}$, where N is the number of cores. Several works [6, 14, 27] have shown that execution suffers significant slowdowns as a result of high communication latency. In response to this, high-radix topologies have been proposed that reduce the network diameter, but at the cost of non-trivial router chip area and energy consumption.

Wireless technology can reduce the latency of a chip-wide communication to a few clock cycles regardless of the size of the chip or the number of cores. To that end, one antenna and transceiver are co-integrated in each core or cluster of cores, as shown in Fig. 1. Information coming from the cores is modulated and the resulting signals propagated through the chip package, bouncing off the

This work was funded in part by NSF Grant No. CCF-1629431 and by EU's Horizon 2020 research and innovation programme Grant No. 863337 (WiPLASH).

metallic heat sink and reaching the receivers. Propagation causes signals to be attenuated a few tens of dBs, mainly due to spreading loss and the relatively high transmission loss in the bulk silicon [25]. These losses, on the other hand, prevent the enclosing package from acting as a reverberation chamber.

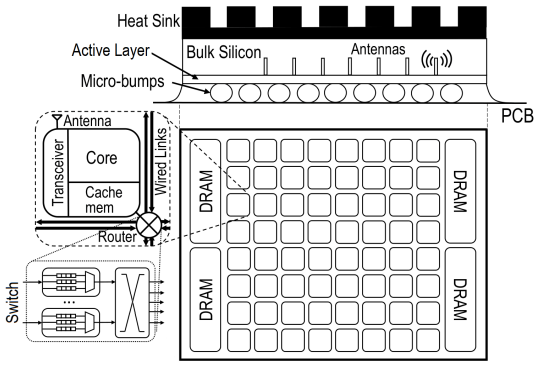


Fig. 1: Wired NoC mesh within a conventional flip-chip package, augmented with one vertical monopole antenna and transceiver per core.

Antennas are either variants of planar integrated dipoles [8] or vertical monopoles implemented with vias that are drilled through the silicon die [25]. In both cases, the operating frequency is chosen within the mm-wave and sub-THz spectrum to minimize antenna size and avoid undesired near-field coupling. The link budget is generally determined by targeting a Bit Error Rate (BER) similar to that of on-chip wires, this is, below 10^{-12}. For the typical channel attenuation values seen in the on-chip environment, it has been shown that simple modulations such as On-Off Keying (OOK) are able to minimize power consumption (toward 1 pJ/bit/cm) and chip area (toward 0.1 mm^2 per transceiver) while maintaining relatively high speeds (10+ Gb/s) [28]. This is because these modulations avoid power-hungry components such as Phase-Locked Loops (PLLs).

From the MAC design perspective, WNoCs are a very demanding scenario. Chip traffic is highly heterogeneous, meaning that the injection rate, burstiness (temporal injection distribution) and hotspotness (spatial injection distribution) of the network vary across and within applications [23]. These characteristics are generally detrimental to performance [5]. On the other hand, the WNoC scenario has some good traits. For instance, the number of nodes is fixed and known beforehand, and due to the enclosed nature of the on-chip scenario, all nodes are within the same transmission range. Therefore, the hidden/exposed terminal effects can be avoided and collisions can be detected [4].

III. The Fuzzy Token Protocol

Fig. 2 illustrates the basic operation of Fuzzy Token. It has two operation modes: *focused* token mode and *fuzzy* token mode. During a focused period, only the token holder can transmit. If the token holder transmits, it is guaranteed to suffer no collision. If the token holder does not transmit, then the protocol switches to fuzzy mode. During a fuzzy period, the token holder can not transmit, but the nodes within what we call the *Fuzzy Area* can transmit. If only one of these nodes attempts to transmit, it succeeds. If more than one of these nodes attempts to transmit, a collision is detected using the NACK mechanism from [4]. In this case, the protocol switches to focused mode. By switching between the two modes, the protocol aims to take advantage of the capabilities of a fair and collision-free token passing protocol (*focused* mode), which works well for high, bursty, and all-core (i.e., uniformly distributed in space) traffic; and of a contention-based protocol (*fuzzy* mode) that performs better for low and few-core (i.e., hotspot) traffic.

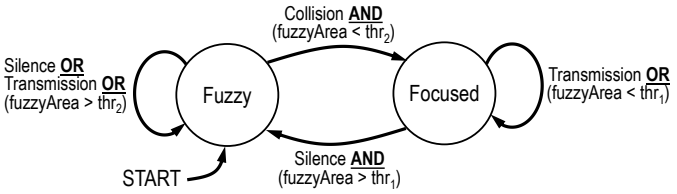


Fig. 2: Basic state diagram of Fuzzy Token.

The amount of contention during a fuzzy period is controlled with the fuzzy area. The fuzzy area includes the nodes around the token holder that may be able to transmit during the period. The idea is to increase the fuzzy area when the load is low to give rapid access to the few nodes that want to transmit, and decrease it otherwise to minimize collisions. To this end, we always increase the fuzzy area when a silence is detected and decrease it when there is a collision. Note that a switch in mode involves a change in the size of the fuzzy area.

Fig. 3 (left side) shows the three areas of Fuzzy Token operation, as a function of the fuzzy area size and the load. When the fuzzy area size is between two given thresholds thr_1 and thr_2, Fuzzy Token follows the state diagram of Fig. 2 (ignore the conditions in parenthesis). We call this area *Normal* in the figure. However, in extreme cases, it is advisable to remain in one of the modes. Specifically, when, after many collisions, the fuzzy area is below thr_1, the network benefits from remaining in the focused mode. On the contrary, when, after many silences, the fuzzy area is over thr_2, the load is low enough for the network to remain in the fuzzy mode.

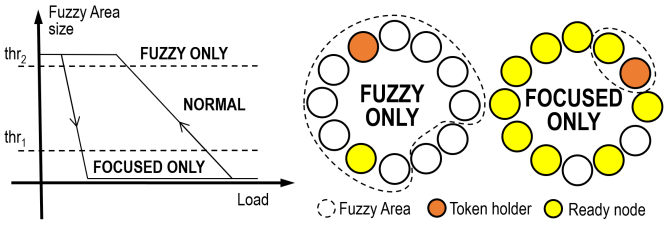


Fig. 3: Transitions (left) and extreme cases (right) of Fuzzy Token.

Fuzzy Token ensures fairness by circulating the token around the virtual ring. In more detail, the token is passed implicitly at every event (silence, collision, or successful transmission) regardless of the protocol mode. This is important because multithreaded applications generally run as slow as the slowest of the threads. Thus, latency tails generated by unfair access will significantly slow down computation, even if they are infrequent. It is worth noting that, thanks to the unique characteristics of the on-chip scenario, all nodes have a consistent view of all events. This allows Fuzzy Token to pass the token implicitly and to update the fuzzy area position and size without explicit messages or centralized control.

A. Algorithm

The algorithm is divided into steps. In each step, all the nodes in the chip are in the same mode (focused or fuzzy), and one node is the token holder. A step lasts for the duration of an event (silence, collision, or successful transmission). On termination of the step, the

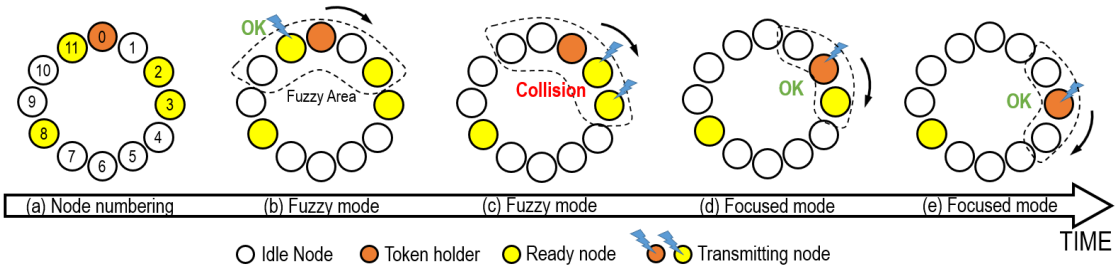


Fig. 4: FUZZY TOKEN protocol example. It assumes the protocol starts in a fuzzy mode step and its initial fuzzy area size is 5.

next node in the virtual ring becomes the token holder, the fuzzy area changes in position and maybe in size, and the mode may change.

The algorithm is executed in each of the nodes of the chip. First, the node checks whether the mode is fuzzy or focused. To this end, there is a bit called *periodMode* that tells the mode of the step. Based on the mode, the operation of the node is as follows:

Fuzzy mode. In fuzzy mode, if the node is in the fuzzy area and has a ready packet, it considers transmitting it with a certain probability. The fuzzy area is encoded with set bits in a vector with as many entries as nodes in the chip. The vector also contains the transmission probability p_i for each node i. If the node decides to transmit, a variant of the BRS protocol [4] is used: the node first sends the preamble and then senses the channel. If the token holder detects a collision of preambles from multiple nodes, it sends a NACK. When a sender senses the NACK, it cancels the transmission; otherwise, it proceeds with the rest of the transfer. By using this mechanism, FUZZY TOKEN reduces the time and energy penalty of collisions, as it avoids the unnecessary transmission of the whole message. The protocol differs slightly from that of [4]: here, it is the token holder the one that sends the NACK, saving energy — in BRS, all idle nodes detect the collision and send NACKs concurrently. For this to work, however, the token holder cannot try to send a packet.

After the packet is sent (C cycles), or the collision is detected (2 cycles), or no packet transmission is even attempted (1 cycle), the algorithm starts a new step. Since the packet size and wireless bitrate are known and static, all nodes are synchronized and proceed in lockstep.

Focused mode. The focused token mode is simpler, as it is based on a conventional token protocol. The token holder is stored in the variable *tokenID*, which is updated at the end of each step. In this mode, the node checks whether it is the token holder and whether it has a packet ready to transmit. If both conditions are true, the node transmits the packet; otherwise, it remains idle. After the C cycles of a successful transmission or 1 cycle of silence, a new step starts.

Housekeeping. At the end of each step, the variables *tokenID* and *fuzzy area* are updated, and *periodMode* may be updated. The *tokenID* is incremented by one modulo the number of nodes in the ring. The fuzzy area is rotated one position as well. In addition, the fuzzy area size is increased if a silence occurred and decreased if a collision occurred. If a successful transmission occurred, we choose not to change the fuzzy area size, as it may be close to the optimal value. Finally, the *periodMode* bit may be updated according to the state machine rules shown in Fig. 2.

B. Design Decisions

FUZZY TOKEN involves several design decisions, and includes a set of parameters that determine how aggressive the protocol is — e.g., how eagerly it changes the fuzzy area size. In this section, we discuss these design decisions. Default values are given in Sec. IV.

Probability of transmission. In fuzzy mode, the probability of transmitting may be a function of the fuzzy area size and the distance to the token holder.

Fuzzy area size update. FUZZY TOKEN uses an additive-increase-multiplicative-decrease (AIMD) approach to update the fuzzy area size. This approach increases the fuzzy area size slowly when the load decreases and reduces it abruptly when collisions are detected. This is because collisions are more harmful than lost opportunities to transmit. Such AIMD approach has been shown to lead to high performance and stability in common protocols. In our scheme, we update the fuzzy area immediately after silence or collision events occur. This is in contrast to protocols that modify their behavior after collecting metrics during an observation epoch (see Sec. VI).

Activation thresholds. In the extremes, FUZZY TOKEN becomes purely driven by the token passing protocol (focused mode) or the contention-based protocol (fuzzy mode). Fig. 3 (right side), shows an example of the two extremes. The figure assumes that thr_1 is 3 nodes and thr_2 is all the nodes in the chip minus 3. In this case, the chart where the fuzzy area includes all the nodes minus 2 is operating in fuzzy mode only. The chart where the fuzzy area includes only two nodes is operating in focused mode only. By setting the thr_1 and thr_2 thresholds, we affect how often the protocol stays in either extreme mode.

Token passing order. A statically-ordered virtual ring is just one of the possible ways of ordering the passing of the token. In any ordering, two hotspot nodes placed close together may lead to multiple collisions and multiple silences. To alleviate this, the token passing order can be changed at runtime, possibly in a pseudo-random way and at every collision. To ensure that all nodes agree on the same order without having to exchange messages, a synchronized random number generator seed can be assumed.

C. Walkthrough Example

Fig. 4 shows an example of FUZZY TOKEN's operation. Fig. 4(a) shows the node numbering, that Node 0 is the token holder, and that Nodes 2, 3, 8 and 11 want to transmit. We assume an initial fuzzy area of 5 and that the protocol is in a fuzzy mode step. Fig. 4(b) shows the operation in the step. Nodes 3 and 8 are outside the fuzzy area, whereas Nodes 2 and 11 are in and could contend for the channel. Assume that only Node 11 proceeds to transmit. Because the transmission is successful, the next step will start with the same fuzzy area size and, following Fig. 2, in fuzzy mode. Fig. 4(c) shows the next step, where the token is owned by Node 1 and the fuzzy area has rotated. Both Nodes 2 and 3 are within the fuzzy area. Assume that both proceed to transmit and therefore collide. As a result, the

fuzzy area size will decrease in the next step. We use an AIMD approach that sets $FA_{new} = \lceil FA_{old}/2 \rceil$, where FA is the fuzzy area size. Assume that the new fuzzy area size is less than thr_2. As a result, the next step will start with a fuzzy area size of 3 and, following Fig. 2, in focused mode. Fig. 4(d) shows the operation in the step. The token holder (Node 2) transmits successfully. As a result, both the fuzzy area size and the focused mode remain unchanged. Fig. 4(e) shows the operation of the next step, where Node 3 is the token holder. In this step, Node 3 transmits successfully.

IV. Simulation Environment

We compare through simulations the average packet latency and throughput of three different protocols: BRS [4], token passing, and FUZZY TOKEN. For a fair comparison, we optimize the token passing protocol with the same assumption as in FUZZY TOKEN, namely, that all nodes have a consistent view of the wireless channel. Thus, the passing of the token is done implicitly, with zero delay.

Evaluations are carried out with the cycle-level Multi2sim [26] architecture simulator. We augmented Multi2sim with wireless on-chip communication modules that model collisions and MAC protocols. Multi2sim admits synthetic traffic and multithreaded applications. The architecture parameters are summarized in Table I. A wireless transfer can be performed in four clock cycles: one for the preamble and three for the payload. BRS [4] and the fuzzy mode of FUZZY TOKEN use one extra cycle to detect a potential NACK.

A. Traffic Patterns

1) Synthetic Traffic Model: Each of the cores acts as a generator that injects packets into the network. Typically, NoCs are evaluated with synthetic traffic models that have, as the main parameter, the injection rate λ in packets/cycle. Common models assume a Poisson process with the same average injection rate for all cores. However, in parallel applications, packet injections show a clear self-similarity caused by the data dependencies between threads of the application. Moreover, common memory patterns such as producer-consumer lead to some cores transmitting more often than others. Our traffic model takes these aspects into account.

To account for the effect of self-similarity, we model a heavy-tailed distribution of traffic via a Pareto distribution [17]. The value of the Hurst exponent is $H \in [0.5, 1)$, which leads to increasing degrees of self-similarity when approaching 1 [23]. Moreover, to model an uneven injection of traffic across nodes, we make use of the hotspotness parameter σ proposed in [23]. There, it was demonstrated that the spatial injection distribution can be approximated as a Gaussian process where the value of σ represents the standard deviation. Low values of σ represent higher concentrations of traffic around a few selected nodes.

2) Real Applications: We model a wireless architecture in Multi2sim. This allows us to obtain the latency statistics of the three MAC protocols when executing multithreaded applications. Table I shows the details of the manycore architecture modeled. We model Replica [13] due to (1) its large speedups with respect conventional architectures, and (2) the relatively high load that it puts on the wireless network. The NoC and memory parameters are in line with the state of the art in manycore processor design. We run a selection of shared-memory multithreaded applications from the Splash2, Parsec, and Crono benchmark suites, which are suited to the characteristics of WNoCs.

TABLE I: Simulated architecture.

Architecture	
Processor chip	64 cores, x86 ISA, 1 GHz, 22-nm tech
L1 I+D	private, 32-KB, 2-way, 64B lines, 2 cycles
L2	shared, 512-KB/core, 8-way, 6 cycles
Coherence	MOESI directory + Replica [13]
Off-chip memory	4 controllers, 100 cycles delay
NoC	2D Mesh, 2 cycles/hop, 128-bit links
WNoC Parameters	
Network	64 nodes, 80-bit packets (preamble: 20 bits)
MAC protocols	BRS [4], Token, FUZZY TOKEN
PHY layer	OOK, 20 Gb/s, power: 39 mW (TX), 39 mW (RX) [13]

B. Performance Metrics

The MAC protocols are evaluated on the basis of packet latency and throughput. Latency is defined as the time between the launching of a packet into the WNoC and its correct reception at all the intended destinations, measured in clock cycles. Throughput is measured in transmitted bits per clock cycle. Another important metric is the energy consumed per transmitted bit, which may increase due to collisons. We calculate the energy per transmitted bit E_{bit} as

$$E_{bit} = E_{OK}\left(1 + \frac{L_{pre}}{L_{tx}} N_{re}\right), \quad E_{OK} = E_{TX} + (N-1)E_{RX}, \qquad (1)$$

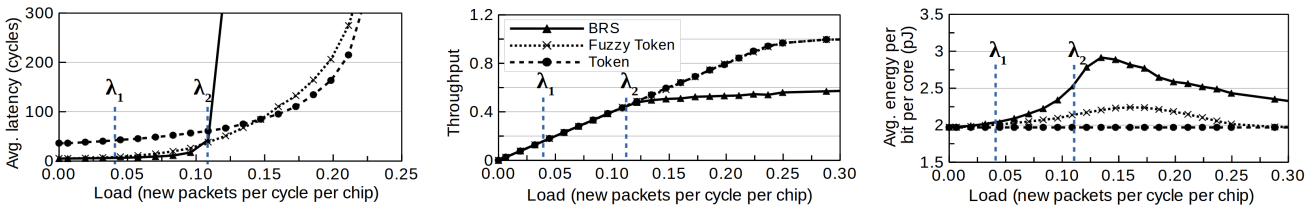
where L_{pre} and L_{tx} are the length of the preamble and of the complete packet, respectively, and N_{re} is the average number of re-transmissions per successfully transmitted packet. E_{OK} is the energy per bit of a non-colliding packet. It can be found as a function of the energy per bit of the transmitting and receiving part of the wireless transceiver (E_{TX} and E_{RX}, respectively), and the number of cores N. In turn, if P_{TX} and P_{RX} are the power consumed by the wireless transmitter and receiver, respectively, and R is the transmission speed, we compute $E_{TX} = P_{TX}/R$ and $E_{RX} = P_{RX}/R$.

V. Evaluation

A. Evaluation with Synthetic Traffic

We start by comparing the protocols with synthetic traffic. By default, arrivals are distributed Poisson and the injection process is equidistributed across all cores. The default FUZZY TOKEN configuration is: equal $1/k$ probability of transmission for all the k nodes inside the fuzzy area, increment the fuzzy area by 1 in each silence, decrement the fuzzy area to half (with ceiling) in each collision, $\text{thr}_1 = 0.1 \times N$, $\text{thr}_2 = 0.9 \times N$, and static token ring. In all cases, we repeat the simulations 10 times and obtain the geometric mean of all runs. Although we performed a sensitivity analysis for many different combinations of these parameters, including a Gaussian probability of transmission function, a wide range of fuzzy area increase/decrease factors, several pairs of thresholds, and a pseudo-random ring ordering, the choices shown in this section were found to be optimal. Due to space constraints, we do not show the sensitivity analyses.

Fig. 5 shows the packet latency (a), packet throughput (b), and energy per bit and core (c) for the different MAC protocols as the load increases. Fig. 5(a) shows that FUZZY TOKEN is able to deliver the low latency of BRS at low loads and almost match the latency of Token at high loads. At intermediate loads, FUZZY TOKEN can outperform both BRS and Token thanks to its intelligent management of contention. In Fig. 5(b), we see that FUZZY TOKEN achieves the same throughput as Token, leaving BRS behind. In fact, at very high loads, FUZZY TOKEN ends up converging to Token by design. We finally



(a) Latency (b) Throughput (c) Energy

Fig. 5: Performance and energy comparison for different MAC protocols over increasing load.

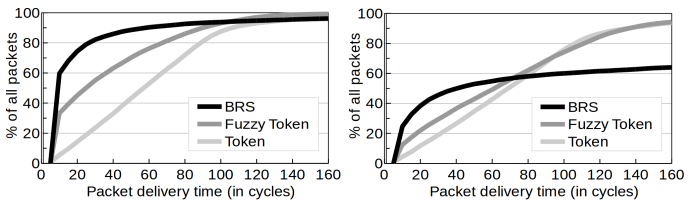
note that, as Fig. 5(c) illustrates, FUZZY TOKEN achieves high performance with only a moderate energy overhead over Token (less than 12%). The overhead is mainly due to collisions at intermediate loads.

Energy-wise, it is also worth noting that the energy consumption of Token is not affected by the load, since the token is passed implicitly, and thus does not consume energy. This, however, comes at the cost of high latency at low loads. In contrast, the energy of BRS increases with the load because of increasing collisions, but then decreases. This effect is due to the finite population of the chip scenario: at very high loads, the backoff reaches high values and reduces the probability of collisions at the expense of unacceptable latency. FUZZY TOKEN attains the low-load latency of BRS while avoiding its high energy consumption at higher loads.

Fig. 6 shows the latency distribution of the three protocols for two different loads: a moderate one (λ_1=0.045 packets per cycle per chip) and an intermediate one (λ_2=0.110 packets per cycle per chip). These loads are also shown in Fig. 5.

At λ_1=0.045 (Fig. 6(a)), most transmissions take less than 30 cycles with BRS, less than 60 with FUZZY TOKEN, and less than 90 with Token. However, due to collisions, 1.29% of the packets with BRS take more than 500 cycles, with a worst-case of ~3400 cycles. On the other hand, FUZZY TOKEN has a worst-case of ~330 cycles, which is the best among the three protocols.

At λ_1=0.110 (Fig. 6(b)), BRS still delivers many packets within the first 60 cycles. However, due to the high number of collisions, 28.9% of the packets take more than 500 cycles to be delivered, with a worst-case of ~110,000 cycles. On the other hand, FUZZY TOKEN also delivers many packets within the first 60 cycles, while providing a worst-case latency of ~390 cycles (again the best among the three protocols). Overall, these plots show the difference between BRS and FUZZY TOKEN. At moderate loads, BRS does well but, at intermediate loads, BRS generates a long tail. In contrast, FUZZY TOKEN approaches the best of BRS and Token at all loads.



(a) Moderate load, $\lambda_1 = 0.045$. Tail: 1.29% in BRS, 0% in others.

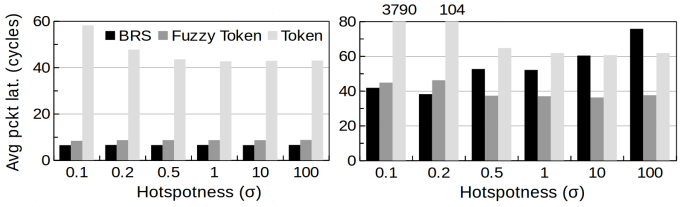
(b) Intermediate load, $\lambda_2 = 0.110$. Tail: 28.9% in BRS, 0% in others.

Fig. 6: Latency cumulative distribution function for the protocols at different loads. Tail is defined as delivery time over 500 cycles.

1) Hotspot Traffic: In hotspot workloads, a few processors inject most of the traffic. In this situation, it has been shown that contention-based protocols such as BRS outperform more rigid collision-free alternatives such as Token [18]. To confirm the hypothesis that FUZZY TOKEN approaches the best of the two types of protocols, we increase the spatial concentration of the traffic via the σ parameter mentioned in Section IV-A1 (i.e., low σ means more hotspot traffic). The inter-arrival time is kept exponential.

Figs. 7(a) and 7(b) show the latency for hotspot traffic with different σ values, at $\lambda_1 = 0.045$ and $\lambda_2 = 0.110$ packets per cycle per chip, respectively. With $\lambda_1 = 0.045$, the load is moderate and contention is low. We show that FUZZY TOKEN is just a couple of cycles slower than BRS, which maintains a very low latency regardless of the value of σ. This is because FUZZY TOKEN has a large fuzzy area and, therefore, nodes can transmit as soon as they generate the packets, irrespective of their location. Token has a high latency, which worsens for low σ because the few transmitting nodes have to wait for their turn to issue a packet.

With $\lambda_1 = 0.110$ (Fig. 7(b)), contention starts to become important. In this case, BRS still benefits from a low number of contenders at small σ, as traffic becomes more concentrated. Token performs poorly in such a situation because many clock cycles are wasted passing the token between a few hotspot nodes. FUZZY TOKEN is capable of maintaining a low latency across all situations, outperforming the two other protocols in nearly all cases. Similar tendencies are observed for loads beyond λ_2, but are not shown for brevity.



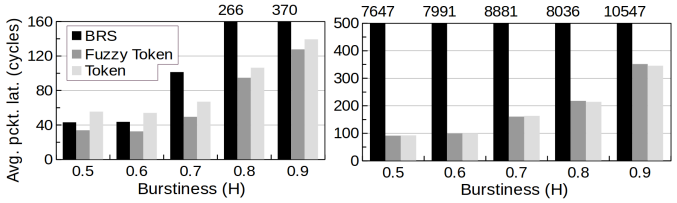
(a) Moderate load, $\lambda_1 = 0.045$ (b) Intermediate load, $\lambda_2 = 0.110$

Fig. 7: Latency for hotspot traffic with different σ values. Lower σ means that fewer nodes inject most of the traffic.

2) Bursty Traffic: We repeat the same set of experiments now changing the temporal distribution of traffic, assuming $\sigma = 100$. Burstiness is modeled via the Hurst exponent H mentioned in Sec. IV-A1, with higher H values leading to longer bursts and longer intervals between bursts.

Figs. 8(a) and 8(b) show the packet latency for different H values, at $\lambda_1 = 0.045$ and $\lambda_2 = 0.110$ packets per cycle per chip, respectively. We observe that burstiness is detrimental for most mechanisms, especially for contention-based protocols like BRS. This is already patent at low loads: bursty injections create collisions. The latency of Token also increases, but its collision-free nature absorbs the bursts better. FUZZY TOKEN is capable of achieving the best performance at all burstiness levels. This is because the first collisions occurring at the burst onset make FUZZY TOKEN to become pure token passing. As the burst is being served, the fuzzy area grows gradually and allows the last nodes of the burst to access the channel earlier.

Fig. 8(b) serves to confirm that increasing the load leads to early saturation, especially for BRS. FUZZY TOKEN avoids contention and converges to Token to better absorb the intense bursty traffic.



(a) Moderate load, $\lambda_1 = 0.045$ (b) Intermediate load, $\lambda_2 = 0.110$

Fig. 8: Latency for different H values. High H means longer bursts.

B. Evaluation with Real Applications

Fig. 9 shows the average packet latency for different MAC protocols normalized to that of FUZZY TOKEN in real applications. This figure showcases the strengths of FUZZY TOKEN, which consistently provides a latency that is among the lowest of the three protocols. On average, FUZZY TOKEN's latency is 4.4× lower than BRS', and 2.6× lower than Token's.

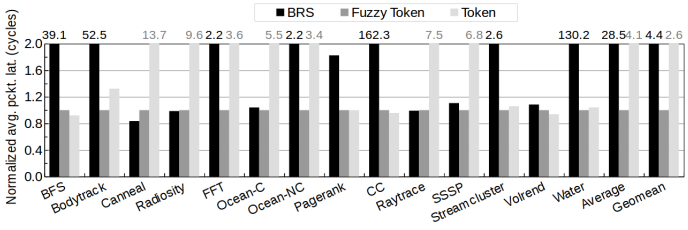


Fig. 9: Average packet latency normalized to FUZZY TOKEN's latency for real applications.

Applications such as *canneal* and *radiosity* have a relatively low load and, therefore, Token performs poorly. Other applications such as *water* and *CC* are communication-intensive and inherently bursty. As a result, BRS suffers an average latency that is two orders of magnitude higher than that of FUZZY TOKEN. In cases such as *bodytrack* and *ocean-nc*, FUZZY TOKEN outperforms both BRS and Token, since traffic alternates between different types of patterns where FUZZY TOKEN is consistently better.

VI. RELATED WORK

Multiplexing: The first MAC protocols proposed for the WNoC paradigm used time-, frequency-, or code-division multiplexing [11, 16]. Those approaches, however, do not scale well beyond a few cores due to the hardware overhead.

Token passing: Different ways of performing token arbitration have been examined [6, 19]. They work well for distributed, high loads, but not for heterogeneous or hotspot traffic. Also, the protocols are not adaptive and do not scale. Mansoor *et al.* attempt to minimize these issues with a predictive scheme that estimates the optimal token occupancy of each node [19]. Duraisamy *et al.* [12] propose a token-like distributed arbitration protocol with single-bit concurrent probing. However, the probing introduces unfeasible bit-level synchronization requirements. FUZZY TOKEN does not need such complex hardware while still leveraging the advantages of token passing at high loads.

Contention-based protocols: They have been explored for WNoCs [4, 18] due to their low latency at low loads. At high loads, however, they saturate early due to collisions. FUZZY TOKEN borrows concepts from BRS-MAC [4], but builds a hybrid protocol that reacts before collisions become too costly.

Hybrid approaches: In [13, 18], token/contention hybrid protocols are proposed to leverage the benefits of both approaches. In [18], utilization metrics are gathered and used to switch between token or random access modes, whereas [13] decides which protocol to use based on the load observed during the first thousands of execution cycles of an application. Instead of working as two discrete protocols connected by a controller, FUZZY TOKEN represents a continuous solution that naturally adapts to the load, for all kinds of workloads.

VII. CONCLUSIONS

This paper has presented FUZZY TOKEN, a MAC protocol uniquely tailored to the particularities of the WNoC scenario. We have shown that with a set of simple rules, FUZZY TOKEN can achieve the low latency of contention-based protocols at low loads and the high throughput of fair collision-free protocols such as token passing at high loads. We have evaluated our protocol in a variety of synthetic traffic patterns and with real application traces, demonstrating 4.4x and 2.6x average reductions in packet latency relative to other state-of-the-art protocols.

REFERENCES

[1] *AMD Epyc 7742 Processor*. https://www.amd.com/en/products/cpu/amd-epyc-7742.

[2] *Ampere Altra 64-Bit Multi-Core ARM Processor*. https://amperecomputing.com/altra/.

[3] *Intel Xeon Platinum 9282 Processor*. https://www.intel.com/content/www/us/en/products/processors/xeon/scalable/platinum-processors/platinum-9282.html.

[4] S. Abadal, *et al.* WiSync: An Architecture for Fast Synchronization through On-Chip Wireless Communication. In *Proceedings of ASPLOS '16*, pages 3–17, 2016.

[5] S. Abadal, *et al.* Medium Access Control in Wireless Network-on-Chip: A Context Analysis. *IEEE Commun. Mag.*, 56(6):172–178, 2018.

[6] S. Abadal, *et al.* OrthoNoC: A Broadcast-Oriented Dual-Plane Wireless Network-on-Chip Architecture. *IEEE Trans. Parallel Distrib. Syst.*, 2018.

[7] J. L. Abellán, *et al.* Electro-Photonic NoC Designs for Kilocore Systems. *ACM J. Emerg. Tech. Com.*, 13(2), 2016.

[8] H. M. Cheema and A. Shamim. The Last Barrier: On-Chip Antennas. *IEEE Microw. Mag.*, 14(1):79–91, 2013.

[9] G. Chen, *et al.* A 340 mV-to-0.9 v 20.2 Tb/s Source-Synchronous Hybrid Packet/Circuit-Switched 16 × 16 Network-on-Chip in 22 nm Tri-Gate CMOS. *IEEE J. Solid-State Circuits*, 50(1):59–67, 2015.

[10] S. Das, *et al.* Optimizing 3D NoC Design for Energy Efficiency: A Machine Learning Approach. In *Proceedings of ICCAD '15*, 2015.

[11] D. DiTomaso, *et al.* A-WiNoC: Adaptive Wireless Network-on-Chip Architecture for Chip Multiprocessors. *IEEE Trans. Parallel Distrib. Syst.*, 2015.

[12] K. Duraisamy, *et al.* Enhancing Performance of Wireless NoCs with Distributed MAC Protocols. In *Proceedings of ISQED '15*, 2015.

[13] V. Fernando, *et al.* Replica: A Wireless Manycore for Communication-Intensive and Approximate Data. In *Proceedings of ASPLOS '19*, 2019.

[14] T. Krishna, *et al.* Towards the Ideal On-chip Fabric for 1-to-Many and Many-to-1 Communication. In *Proceedings of MICRO-44*, 2011.

[15] H. Kwon, *et al.* A Communication-Centric Approach for Designing Flexible DNN Accelerators. *IEEE Micro*, 38(6):25–35, 2018.

[16] S.-B. Lee, *et al.* A Scalable Micro Wireless Interconnect Structure for CMPs. In *Proceedings of MOBICOM '09*, 2009.

[17] W. E. Leland, *et al.* On the Self-Similar Nature of Ethernet Traffic. *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[18] N. Mansoor and A. Ganguly. Reconfigurable Wireless Network-on-Chip with a Dynamic Medium Access Mechanism. In *Proceedings of NOCS '15*, 2015.

[19] N. Mansoor, *et al.* A Demand-Aware Predictive Dynamic Bandwidth Allocation Mechanism for Wireless Network-on-Chip. In *Proceedings of SLIP '16*, 2016.

[20] H. Mondal, *et al.* Interference-Aware Wireless Network-on-Chip Architecture using Directional Antennas. *IEEE Trans. Multi-Scale Comput. Syst.*, 2017.

[21] G. P. Nychis, *et al.* On-Chip Networks from a Networking Perspective: Congestion and Scalability in Many-Core Interconnects. In *Proceedings of SIGCOMM '12*, 2012.

[22] D. Sánchez, *et al.* An Analysis of On-Chip Interconnection Networks for Large-Scale Chip Multiprocessors. *ACM T. Archit. Code Op.*, 7(1), 2010.

[23] V. Soteriou, H. Wang, and L. Peh. A Statistical Traffic Model for On-Chip Interconnection Networks. In *Proceedings of MASCOTS '06*, 2006.

[24] K. Terplan and P. A. Morreale. *The Telecommunications Handbook*. CrC Press, 2018.

[25] X. Timoneda, *et al.* Engineer the Channel and Adapt to it: Enabling Wireless Intra-Chip Communication. *IEEE Transactions on Communications*, 68(5):3247–3258, 2020.

[26] R. Ubal, *et al.* Multi2Sim: A Simulation Framework for CPU-GPU Computing. In *Proceedings of PACT '12*, 2012.

[27] X. Xiang, *et al.* A Model for Application Slowdown Estimation in On-Chip Networks and Its Use for Improving System Fairness and Performance. In *ICCD '16*, 2016.

[28] X. Yu, *et al.* Architecture and Design of Multi-Channel Millimeter-Wave Wireless Network-on-Chip. *IEEE Design & Test*, 31(6):19–28, 2014.