Fuzzy-Token: An Adaptive MAC Protocol for Wireless-Enabled Many-Core CMPs

Paper # 1969

Abstract—**Recent computer architecture trends herald the arrival of massive multiprocessors with hundreds of cores within a single chip. Unfortunately, in this context, traditional on-chip networks have been proven to not scale well neither in terms of latency or energy consumption, leading to a bottleneck in the computation. The Wireless Network-on-Chip (WNoC) paradigm holds considerable promise for the implementation of on-chip networks that will enable such massive multicore chips. One of the main challenges, however, resides in the design of methods that provide fast and efficient access to the wireless channel while adapting to the constant traffic changes within and across applications. Existing approaches are either cumbersome or do not provide the required adaptivity. We propose FUZZY TOKEN, a simple protocol that leverages the unique properties of the on-chip scenario to deliver efficient and low-latency access irrespective of the application characteristics. We substantiate our claim via simulations with a synthetic traffic suite and real application traces. FUZZY TOKEN consistently provides a latency among the lowest of the evaluated protocols. In average, the latency provided by FUZZY TOKEN is** $28.5\times$ **lower than previously proposed WNoC MAC protocols, such as BRS** ($4.4\times$ **in geometric mean), and** $4.1\times$ **lower than Token** ($2.6\times$ **in geometric mean), as evaluated and discussed in depth in this paper.**

I. INTRODUCTION

Recent years have witnessed the emergence of computing architectures that integrate up to a thousand processor cores and memory on a single die. This includes heterogeneous systems with integrated GPUs [19], neural network accelerators, and general-purpose manycore CPUs. Processor cores communicate among them and with memory to share data and synchronize execution. The importance of communication grows with the number of cores, to the point of having become the key performance bottleneck in manycore processors [21, 22].

Modern processors account for Networks-on-Chip (NoCs) to serve inter-core traffic [5, 12]. NoCs are essentially packet-switched networks of on-chip routers and links, typically arranged in a 2D grid. While this approach is more scalable and efficient than buses, the latency and energy of NoCs start to become a problem at hundred cores [21, 22]. This is specially so for broadcast and chip-scale messages that need to traverse a high number of routers before arriving to the intended destination [11]. The increasing latency raises the likelihood of stalling remote processor cores as they wait for results or to synchronize, which throttles execution. Slowdowns of $2\times$-$3\times$ have been estimated in processors with several tens of cores [11, 26] and even worse slacks are expected at higher core counts.

To guarantee the scalability of processor architectures to thousand cores and beyond, emerging interconnect technologies such as 3D [6], nanophotonics [3], or wireless [13] 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 better than that of conventional NoCs by an order of magnitude [2]. Moreover, the lack of path infrastructure between cores provides a network-level flexibility unattainable with other technologies [9].

The wireless approach is enabled by recent advances in CMOS RF technology allowing the on-chip integration of millimeter-wave antennas and transceivers [4, 10, 13, 27]. Based on these, multiple Wireless Network-on-Chip (WNoC) designs have been proposed that achieve substantial network-level improvements with respect to conventional NoCs [2, 14, 20]. More importantly, complete multiprocessor architectures based on the WNoC paradigm have demonstrated execution speedups of up to $10\times$ ($1.89\times$ in average) and reduced the multiprocessor energy consumption by an average of 38% [9]. This has been achieved by considering bandwidths of a few tens of GHz and low-order modulations feasible with current technology.

To fully realize the potential of WNoC, however, it is necessary to build Medium Access Control (MAC) mechanisms able to cope with the heterogeneity and performance requirements of the multiprocessor scenario [1]. MAC protocols need to adapt to wild changes in the on-chip traffic, which can be hotspot or bursty, while incurring little to no delay on the transmission. As analyzed in Section VI, existing efforts [9, 14, 16, 17, 18] are not capable of capturing 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 demands of the WNoC for every application, minimizing transmission latency and increasing network throughput. The FUZZY TOKEN algorithm combines the strengths of token passing and random access and, with a few simple rules, adapts to all types of workloads. The algorithm is evaluated using, first, a synthetic traffic model that emulates a wide variety of applications [23] and, second, a set of real application traces that showcase the potential of FUZZY TOKEN for speeding up execution.

The paper is organized as follows. Section II provides background about NoC and traffic in manycore processors. Section III depicts FUZZY TOKEN. Section IV describes the evaluation methodology and Section V presents the main evaluation results. Related works are analyzed in Section VI and the paper is concluded in Section VII.

II. BACKGROUND

Traditionally, wired NoC architectures use a packet-switched network with each processor connected to a router as shown in Figure 1. Each router has several pipeline stages for enqueuing, route computation, arbitration, and dequeuing of packets [22]. This way, data packets move from source to destination via multiple hops with each hop incurring delays and energy consumption. The mesh topology is the *de facto* standard 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 [2, 11, 26] have shown that computation suffers significant slowdowns as a by-product of high communication latency. In response to this, high-radix topologies have been proposed that reduce the network diameter, but at the cost of a non-trivial chip area and energy cost at the routers.

Wireless technology can allow to reduce the chip-wide latency to a few clock cycles regardless of the size of the chip or the number of cores. To that end, one antenna and transceiver is co-integrated with each core or cluster of cores, as shown in Fig. 1. Information coming from the cores is modulated and the resulting signals propagate through the chip package, bouncing off the 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 dielectric loss at the bulk silicon [24]. These losses, on the other hand, prevent the enclosed package to act as a reverberation chamber.

Antennas are either variants of planar integrated dipoles [4] or vertical monopoles implemented with vias that are drilled through the silicon die [24]. In both cases, the operating frequency is chosen within the mm-wave and sub-THz spectrum to minimize the antenna size and avoid undesired near-field coupling. The link budget is performed generally assuming a target 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 (towards 1 pJ/bit/cm) and chip area (towards 0.1 mm^2 per transceiver) while maintaining relatively high speeds (10+ Gb/s) [10, 13, 27]. This is because these modulations avoid power-hungry components such as Phase-Locked Loops (PLLs).

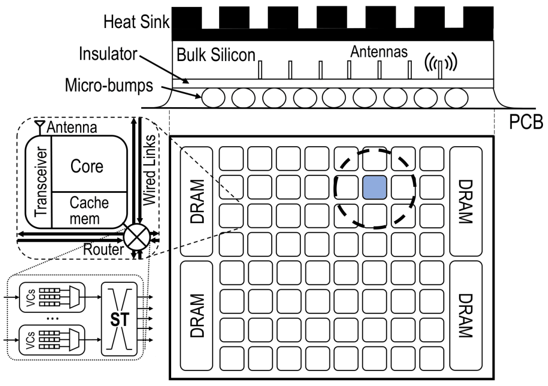


Fig. 1: Wired NoC mesh within a conventional flip-chip package, augmented with one vertical monopole antenna and transceiver per core.

From the MAC design perspective, WNoCs are a very demanding scenario. Traffic chip is highly heterogeneous, meaning that the injection rate, burstiness (temporal injection distribution) and hotspotness (spatial injection distriubtion) of the network vary across applications, and even across the different phases of a single application [23]. These characteristics are generally detrimental to performance [1]. To address these challenges, designers can resort to the uniquenesses of the scenario. 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 even detected [18]. These aspects are leveraged in our design, which is described next.

III. The Fuzzy Token Protocol

Figure 2 illustrates the basic operation of Fuzzy Token. Time is divided into two kinds of periods or modes: *focused* token mode and *fuzzy* token mode. During focused token periods, only the token holder can transmit. If the token holder transmits, the channel is occupied during a number of cycles with guaranteed no collision. If the token holder does not transmit, then the protocol switches to fuzzy token mode. During such a period, the nodes within what we call the *fuzzy area* will contend for the channel. If a node wins the contention, it transmits successfully. Collisions can occur in this period, which are resolved by using the NACK mechanism from [18]. However, colliding nodes do not back off; instead, they just wait for another opportunity to transmit in forthcoming cycles. In the case of a collision, the protocol switches back to the focused token mode. By alternating between the two modes, the protocol aims to take advantage of the capabilities of a fair and collision-free token passing, which works well for high, bursty, and evenly distributed loads; and of a contention-based protocol that performs better for moderate loads and hotspot traffic.

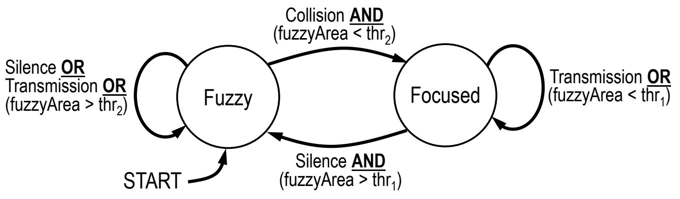

Collision AND (fuzzyArea < thr2)
Silence OR Transmission OR (fuzzyArea > thr2)
Fuzzy
Focused
Transmission OR (fuzzyArea < thr1)
START
Silence AND (fuzzyArea > thr1)

Fig. 2: Basic state diagram of Fuzzy Token.

The amount of contention during the fuzzy token periods is controlled with the fuzzy area. The fuzzy area marks the nodes around the token holder that may be able to transmit in a given cycle. The main 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 increase the fuzzy area when a silence is detected and decrease it when there is a collision. In extreme cases, it is advisable to stop alternating between the focused and fuzzy modes. This is exemplified in Figure 3. When the fuzzy area is below a given threshold after many collisions, the network will benefit from staying in the focused token mode. On the contrary, when the fuzzy area is over a certain threshold after many silences, it is considered that the load is low enough to stay in fuzzy token mode. The normal alternating operation of the protocol is reinstated when the value of the fuzzy area increases/decreases again, respectively.

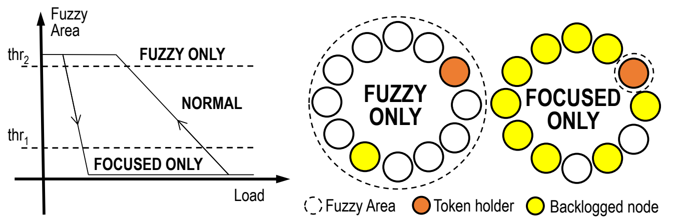


Fig. 3: Transitions (left) and extreme cases (right) of Fuzzy Token.

Fuzzy Token ensures fairness by circulating the token within 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 to pass the token implicitly and to update the fuzzy area values without explicit messages or centralized control.

A. Algorithm

The algorithm is executed in each of the nodes within the chip. First, the node checks whether we are in fuzzy or focused mode. To this end, we have a bit called *periodMode* that is set to ’0’ if the mode is fuzzy, or ’1’ otherwise. After executing one iteration, the internal metrics are updated and the algorithm is executed again. The main procedures are as follows:

Fuzzy token mode. In the fuzzy mode, we first check whether there is a packet ready. If so, the node confirms if it is in the *fuzzy area*,

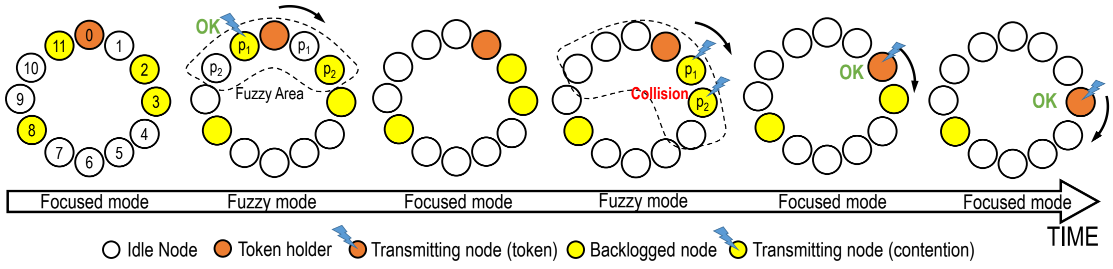


Fig. 4: Basic operation of the protocol. The initial fuzzy area is 5 and $p_1 > p_2$ is assumed.

which is encoded in a vector with as many positions as cores in the chip. The vector also contains the transmission probability p_i for each node i. After deciding whether the node transmits or not, a variant of the BRS protocol mechanisms [18] is used. In transmission, the preamble is sent and the channel is then sensed: the transmission is cancelled upon receiving a NACK or continued otherwise. By using this mechanism, FUZZY TOKEN reduces the time and energy penalty of collisions, as it avoids the unnecessary transmission of the whole message. In reception, the protocol differs slightly from that of [18]: here, only the token holder sends a NACK after detecting a collision, saving energy (in BRS, all idle nodes detected the collision and sent the NACK concurrently). For this to work, however, the token holder cannot participate in the contention, which results in a negligible penalty. Regardless of whether there is a silence, a collision, or a successful transmission, the algorithm will start a new iteration after a number of cycles. Since the packet size and wireless bitrate are known and static, all nodes are synchronized and proceed in lockstep.

Focused token 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 iteration. In this mode, the node checks whether there is a packet ready and transmits it only if the node is the token holder as indicated in the tokenID. Otherwise, it remains idle waiting for incoming transmissions. Both actions lead to a new iteration of the algorithm after the appropriate number of cycles (C after a successful transmission and 1 after a silence).

Housekeeping. At the end of each iteration the variables *tokenID*, *fuzzy area*, and *periodMode* are updated. The *tokenID* is updated based on a circular buffer that stores the token ring order which, by default, uses the coreID in an increasing order. The *fuzzy area* is generally increased in the absence of transmissions (regardless of the mode) and decreased in the presence of collisions. In the event of a successful transmission, we choose to not change the fuzzy area as it may be close to the optimal value. In either case, after updating the *fuzzy area*, the *periodMode* bit is also updated according to the state machine rules shown in Fig. 2 and the hard-coded threshold values. For more details on the possible update policies, we refer the reader to the next sub-section.

B. Design Decisions

FUZZY TOKEN involves several design decisions and includes a set of parameters that determine how aggressive is the protocol and how reactive it is to changes in the workload. Default values are given in Sec. V.

Probability decay. In fuzzy mode, the probability of transmitting is a function of the fuzzy area and the distance to the token holder.

Fuzzy area update. FUZZY TOKEN uses an additive-increase-multiplicative-decrease (AIMD) approach to update the fuzzy area. This opens up the fuzzy area slowly when the load decreases and reduces it abruptly for few collisions. This is because collisions are more harmful than lost opportunities to transmit, and such AIMD approach has been shown to lead to high performance and stability in widespread protocols. In our scheme, we update the fuzzy area after each relevant silence or collision. 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 (focused mode only) or a pure contention-based protocol (fuzzy mode only). As shown in Fig. 3, we set two thresholds thr_1 and thr_2 that determine how often the protocol stays in either extreme mode. This, therefore, determines the behavior of FUZZY TOKEN in intermediate loads.

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 followed by 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 seed can be assumed.

C. Walkthrough Example

Figure 4 shows an example of FUZZY TOKEN's operation. In this case, nodes 2, 3, 8 and 11 want to transmit and we assume a fuzzy area of 5. We start in focused mode with node 0 being the token holder. Since it remains silent, the fuzzy area is increased and the token period starts. Nodes 3 and 8 are outside the fuzzy area, whereas nodes 2 and 11 contend for the channel. Node 11 has higher probability p_1 of transmitting than node 2, p_2, as it is closer to the token holder. We assume that only Node 11 transmits in this case. The token is then passed to node 1, which remains silent during the focused token period. Then, nodes 2 and 3 are within the fuzzy area and contend. We now assume that despite having different probabilities, both attempt to transmit and collide. This makes the fuzzy area to be cut. The token is thus passed to node 2 which transmits with guaranteed no collision during his focused token period. The token is then passed to node 3, which continues in focused token period also transmits successfully.

IV. SIMULATION ENVIRONMENT

A. Simulation Methodology

In order to prove the efficiency of FUZZY TOKEN, we simulate and compare the average packet latency and throughput of three different protocols: BRS [18], a token passing protocol, and our FUZZY TOKEN protocol. For a fair comparison, we optimize the token passing protocol with the same assumption than in FUZZY TOKEN, namely, that all nodes have a consistent view of the wireless channel. Thus, the passing of the token can be made implicitly, with zero delay after transmission and one-cycle delay after silence.

TABLE I: Simulated architecture and applications.

Architecture	
System	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 [9]
Off-chip	4 controllers, 100 cycles delay
NoC	2D Mesh, 2 cycles/hop, 128-bit links
WNoC Parameters	
NET	64 nodes, 80-bit packets (preamble: 20 bits)
MAC	BRS [18], Token, FUZZY TOKEN
PHY	OOK, 20 Gb/s, TX: 39 mW, RX: 39 mW [9]
Applications	
Parsec	bodytrack, streamcluster, canneal
Splash2	FFT, ocean, radiosity, raytrace, volrend, water
Crono	BFS, SSSP, PageRank, CC

Evaluations are carried out in the cycle-accurate architecture simulator Multi2sim [25]. Multi2sim has been augmented with wireless on-chip communication modules that model collisions and the aforementioned MAC protocols. Multi2sim admits synthetic traffic and multithreaded applications. The architecture and application parameters are summarized in Table I. From there, we can infer that the transmission time is four clock cycles. The preamble is set to take a single clock cycle. The NACK delay is also a single cycle, which is used for both BRS [18] and the fuzzy mode of FUZZY TOKEN.

B. *Traffic Patterns*

1) Synthetic Traffic Model: In a multicore setting, each of the processor cores essentially acts a generator that injects packets into the network. Typically, NoCs are evaluated with synthetic traffic models that have, as main parameter, the injection rate λ in packets/cycle. Widespread simple 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 within 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 [15]. The value of $H \in [0.5, 1)$, the Hurst exponent, leads to increasing degrees of self-similarity when approaching to 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 cores.

2) Real Applications: To evaluate the performance of FUZZY TOKEN within a full manycore, we build a wireless architecture inside Multi2sim. This allows to obtain the latency statistics of the three MAC protocols when executing multithreaded applications. Table I shows the details of the manycore architecture modeled for the analysis. We chose Replica [9] due to (1) its large speedups (1.89× on average) with respect conventional architectures and (2) the relatively high load that it casts 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 benchmarks from the Splash2, Parsec, and Crono benchmark suites which are suited to the characteristics of WNoC.

C. *Performance Metrics*

The MAC protocols are evaluated on the basis of packet latency and throughput. Latency is defined as the time between the generation of a packet and its correct reception at all the intended destinations, measured in clock cycles, whereas 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 keep track of the total number of collisions and calculate the energy per 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 a preamble and of the complete packet, respectively, and N_{re} is the average number of re-transmissions per successfully transmitted packet. E_{OK} is energy of a non-colliding message, whereas $E_{TX} = P_{TX}/R$ and $E_{RX} = P_{RX}/R$ are the energy per bit of the transmitting and receiving part of the wireless transceiver, with power P_{TX} and P_{RX} respectively. Also, R is the transmission speed and N is the number of cores. We express the result in pJ/bit/core.

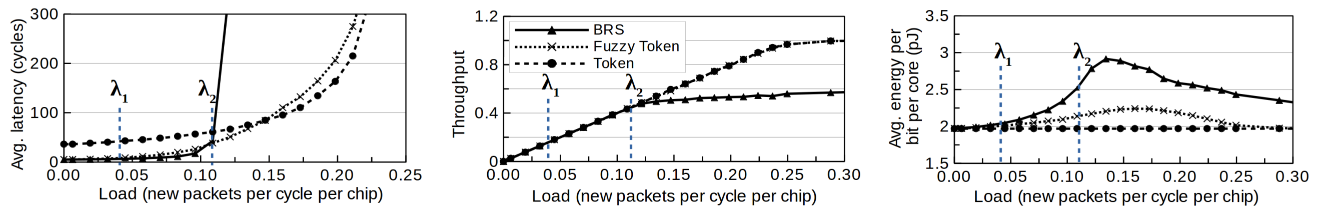
V. EVALUATION

A. *Evaluation with Synthetic Traffic*

We start by comparing FUZZY TOKEN against the baseline protocols with synthetic traffic. By default, arrivals are Poisson and the injection process is equidistributed across all cores. The default FUZZY TOKEN configuration is: no probabilistic decay (full probability), increment the fuzzy area by 1 in each silence, decrement the fuzzy area to 1 in each collision, thr$_1$ = 10%, thr$_2$ = 90%, 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 decay function, a wide range of fuzzy area increase/decrease factors, several pairs of thresholds, and a pseudo-random ring ordering, the ones shown in this section were found to be the optimum. Due to space constraints, we do not show the full comparison against the other configurations tested.

The results of this first analysis are summarized in Figure 5. It is observed in Fig. 5(a) how FUZZY TOKEN is able to deliver the low latency of BRS at low loads and almost match the latency of token at high loads. In intermediate loads, FUZZY TOKEN can even 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 than token, leaving BRS behind. In fact, at very high loads, FUZZY TOKEN ends up converging to regular token passing by design. We finally note that, as Figure 5(c) illustrates, FUZZY TOKEN achieves high performance with a very moderate energy overhead over the regular token passing (less than 12%). The overhead is mainly due to collisions at intermediate loads.

Energy-wise, there are two additional points that are worth remarking. First, the energy consumption in regular token passing is not affected by the load. This is because of the implicit passing of the token, which does not consume energy. This, however, comes at the cost of high latency at low loads. The second point is that the energy of BRS increases with the load because of the appearance of collisions, but then decreases. This effect is due to the finite population of the chip scenario: at very high loads, the backoff reaches huge values and reduces the probability of collisions at the expense of unacceptable

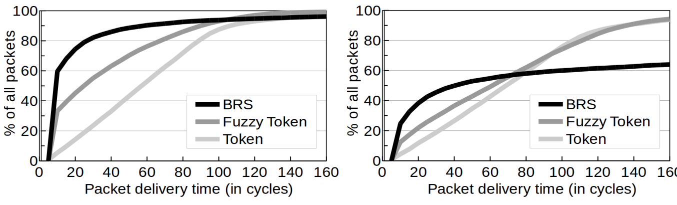


(a) Latency (b) Throughput (c) Energy

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

latency. We see that FUZZY TOKEN achieves the low-load latency of BRS while avoiding its high energy expense at higher loads.

To complete the analysis, we plot the latency distribution of the three protocols in Figure 6 for two different loads. At moderate loads, Figure 6(a), it is observed that most transmissions with BRS take less than 30 cycles, with FUZZY TOKEN less than 60, and with Token less than 90. The caveat, however, is that 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 (the best among the three protocols), while still serving most of the packets within 60 cycles. At intermediate loads, Figure 6(b) shows how BRS still delivers many packets within the first 60 cycles, however now due to the higher amount 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 most of the packets within the first 60 cycles, while providing a worst-case latency of ~390 cycles (again the best among the three protocols). The difference between BRS and FUZZY TOKEN, then, is shown in the re-distribution of the latency. While the latency distribution of FUZZY TOKEN gradually becomes the best between BRS and Token, BRS generates a very long tail that can take values up to several thousand cycles, which would clearly be a bottleneck for the execution of the application.



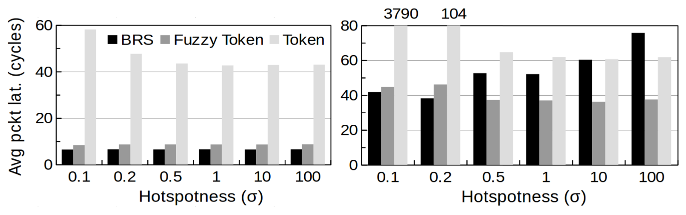
(a) Moderate load, $\lambda_1 = 0.045$. Tail: 1.29% BRS, 0% others.

(b) Intermediate load, $\lambda_2 = 0.11$. Tail: 28.9% BRS, 0% others.

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

1) Hotspot Traffic: In hotspot workloads, a few processors inject most of the traffic. In that situation, it has been shown that contention-based protocols such as BRS outperform more rigid collision-free alternatives such as token [16]. To confirm the hypothesis that FUZZY TOKEN can deliver the best of the two types of protocols, we increase the spatial concentration of traffic via the σ parameter mentioned in Section IV-B1 (i.e. low σ means hotspot traffic). The inter-arrival time is kept exponential.

Figure 7 shows the results of the analysis at $\lambda_1 = 0.045$ and $\lambda_2 = 0.11$ packets/cycle. Fig. 7(a) illustrates the former case, where the load is moderate, contention is low, and readiness is more appropriate. We confirm how 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 nodes can transmit as soon as they generate the packets irrespective of their location. Token has a high latency, which worsens for low σ as the few transmitting nodes have to wait for their turn for each and every packet. Fig. 7(b) shows the results in the latter case, where contention starts becoming important (Fig. 5 shows how BRS starts to saturate at that load). In this case, BRS benefits from the decreasing number of contenders as traffic becomes more concentrated ($\sigma = 0.1$). Token passing performs poorly in such a situation because many clock cycles are wasted passing the token between a few greatly backlogged nodes. FUZZY TOKEN is capable of maintaining a low latency across all situations, outperforming the two other options by upto 100× with respect to token and upto 47% with respect to BRS. It is worth noting that similar tendencies are observed for loads beyond λ_2, but not shown in the sake of brevity.



(a) Moderate load, $\lambda_1 = 0.045$ (b) Intermediate load, $\lambda_2 = 0.11$

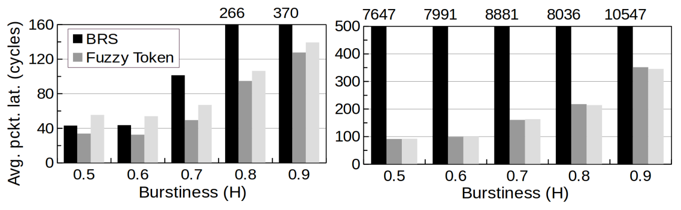
Fig. 7: Latency for hotspot traffic with different σ values. Lower σ means that fewer nodes inject most of the traffic.

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 as depicted in Sec. IV-B1, with higher H values leading to longer bursts and longer intervals between bursts.

Figures 8(a) and 8(b) illustrate the impact of burstiness on the MAC performance at moderate and intermediate loads, respectively. The first observation is that burstiness is detrimental for most mechanisms, especially for contention-based protocols. This is already patent at low loads: injections are infrequent, but bursty, and hence collisions cannot be avoided. The latency of token passing also increases, but its collision- free nature better absorbs the bursts. Even so, 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. Figure 8(b) serves to confirm that increasing burstiness leads to early saturation, especially for BRS. As a result, FUZZY TOKEN avoid contention and converges to token passing to better absorb the intense bursty traffic.

B. Evaluation with Real Applications

Figure 9 shows the speedup in terms of packet latency of FUZZY TOKEN with respect to the other MAC protocols. A value of 2 means that FUZZY TOKEN is twice as fast as the protocol being evaluated. This plot exemplifies the strengths of FUZZY TOKEN, which consistently provides a latency which is among the lowest



(a) Moderate load, $\lambda_1 = 0.045$ (b) Intermediate load, $\lambda_2 = 0.11$

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

of the three protocols. In average, the latency provided by FUZZY TOKEN is 28.5× lower than BRS (4.4× in geometric mean) and 4.1× lower than token (2.6× in geometric mean).

Applications such as *canneal* or *radiosity* have a relatively low load and, there, token performs poorly. Other applications such as *water* or *CC* are communication-intensive and inherently bursty. As a result, BRS shows a latency two orders of magnitude higher than that of FUZZY TOKEN. In cases such as *bodytrack* or *ocean*, FUZZY TOKEN clearly outperforms both BRS and token mainly because traffic alternates between different types of patterns where FUZZY TOKEN is consistently better.

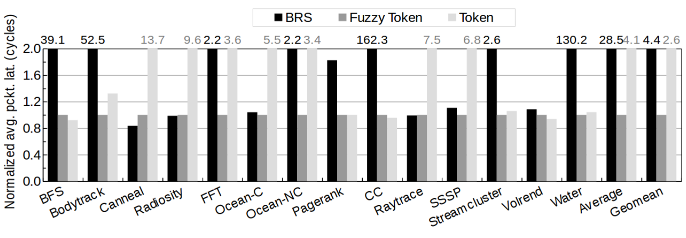


Fig. 9: Average packet latency, normalized to FUZZY TOKEN's latency, for different real application traces.

VI. RELATED WORK

Multiplexing: The first MAC protocols proposed for the WNoC paradigm used time, frequency, or code orthogonal channels [7, 14]. Those approaches, however, do not scale well beyond a few cores due to the hardware overhead. Spatial multiplexing has also been proposed in [20], but the use of directional antennas eliminates the broadcast advantage of WNoC.

Token passing: Different variants of token passing have been examined as alternatives to channelization [2, 17]. These solutions 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 by means of a predictive scheme that estimates the optimal token occupancy of each node [17]. The work of [8] proposes 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. On top of that, FUZZY TOKEN is flexible and scalable.

Random access protocols: contention-based protocols have been explored for WNoC [16, 18] due to their low latency at low loads. BRS-MAC [18] minimizes latency also under moderate loads by using preamble transmission, collision detection, or collision notification via negative ACKs. At high loads, however, the protocol saturates early due to the many collisions. FUZZY TOKEN borrows concepts from BRS-MAC [18], but builds a hybrid protocol that reacts before collisions become too penalizing.

Hybrid approaches: In [9, 16], token/contention hybrid protocols are proposed that attempt to leverage the benefits of both approaches. In [16], utilization metrics are gathered and used to switch between token or random access modes, whereas [9] 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. As shown in Sec. V, this provides finer-grained control and better performance at 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. We have evaluated our protocol in a variety of synthetic traffic patterns and with real application traces, demonstrating a severalfold speedup with respect to other state-of-the-art protocols. Our contribution, together with other advances in the field of on-chip networking, pave the way for scalable and efficient manycore processors for general-purpose computing, machine learning acceleration, and data center as well as high-end server processors.

REFERENCES

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

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

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

[4] H. M. Cheema and A. Shamim. The last barrier: On-chip antennas. *IEEE Microw. Mag.*, 14(1):79–91, 2013.

[5] 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.

[6] S. Das, *et al.* Optimizing 3d noc design for energy efficiency: A machine learning approach. In *Proceedings of the ICCAD '15*, 2015.

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

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

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

[10] D. Fritsche, *et al.* A Low-Power SiGe BiCMOS 190-GHz Transceiver Chipset With Demonstrated Data Rates up to 50 Gbit/s Using On-Chip Antennas. *IEEE Trans. Microw. Theory Techn.*, 65(9):3312–3323, 2017.

[11] T. Krishna, *et al.* Towards the ideal on-chip fabric for 1-to-many and many-to-1 communication. In *Proceedings of the MICRO-44*, 2011.

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

[13] S. Laha, *et al.* A New Frontier in Ultralow Power Wireless Links: Network-on-Chip and Chip-to-Chip Interconnects. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 34(2):186–198, 2015.

[14] S.-B. Lee, *et al.* A scalable micro wireless interconnect structure for CMPs. In *Proceedings of the MOBICOM '09*, 2009.

[15] W. E. Leland, *et al.* On the self-similar nature of Ethernet traffic. *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

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

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

[18] A. Mestres, *et al.* A MAC protocol for Reliable Broadcast Communications in Wireless Network-on-Chip. In *Proceedings of the NoCArc '16*, 2016.

[19] A. Mirhosseini, *et al.* Binochs: Bimodal network-on-chip for cpu-gpu heterogeneous systems. In *Proceedings of the NOCS '17*, 2017.

[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 *Proc. SIGCOMM*, 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.