ETH zürich

Scalable 10 G TCP/IP Stack Architecture for Reconfigurable Hardware

David Sidler, Gustavo Alonso · Dept. of Computer Science, ETH Zürich
Michaela Blott, Kimon Karras, Kees Vissers · Xilinx Research
Raymond Carley · Carnegie Mellon University

Motivation

- Data center applications require a TCP/IP stack supporting thousands of connections
- Most implementations on FPGAs are optimized for low-latency and support only a few connections
- Allows straightforward integration of specialized hardware into existing infrastructure

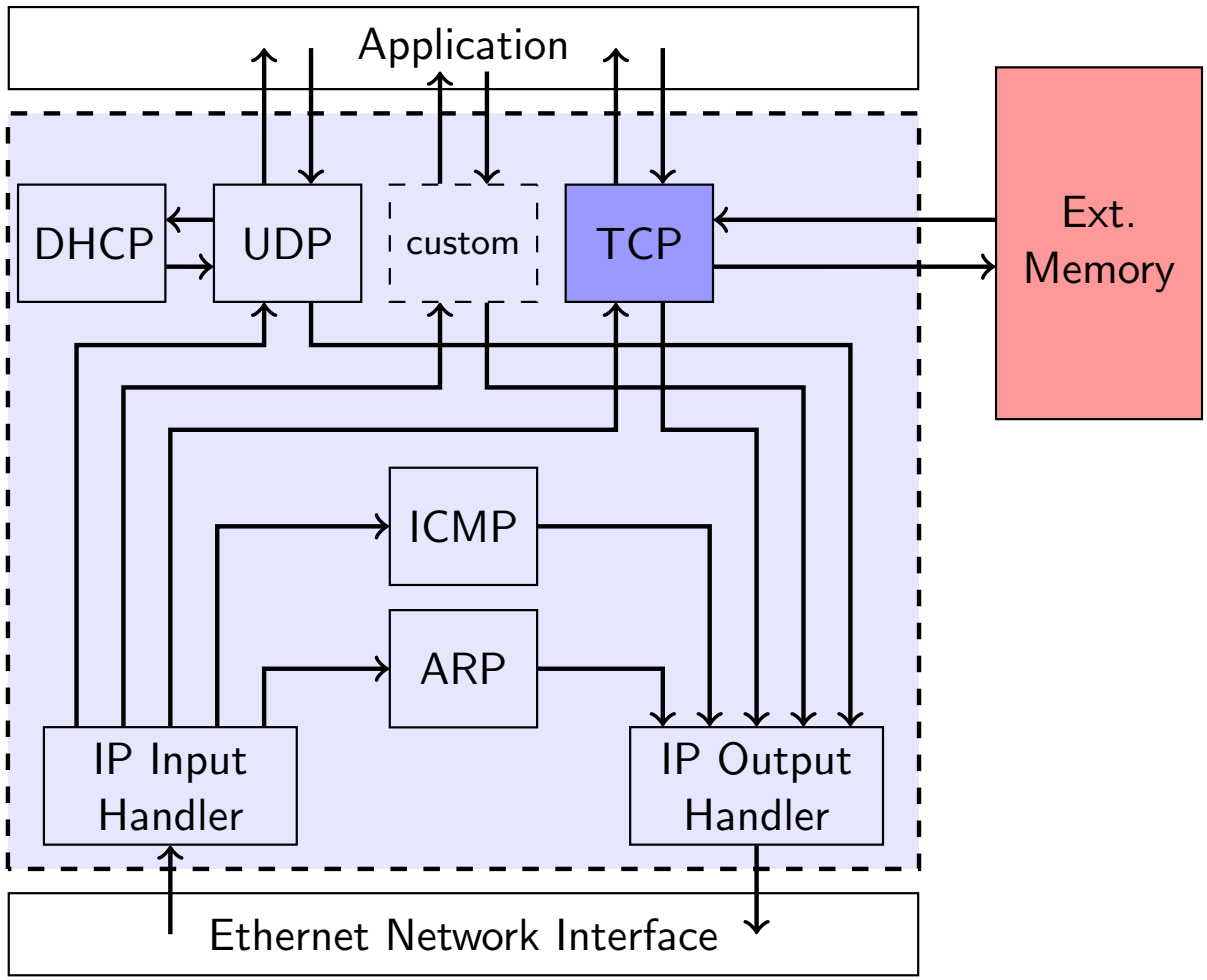
Goal

- 10 Gbps throughput
- Support thousands of concurrent connections
- Scalable and flexible architecture
- Use high-level synthesis (C/C++) to shorten development time

Challenges

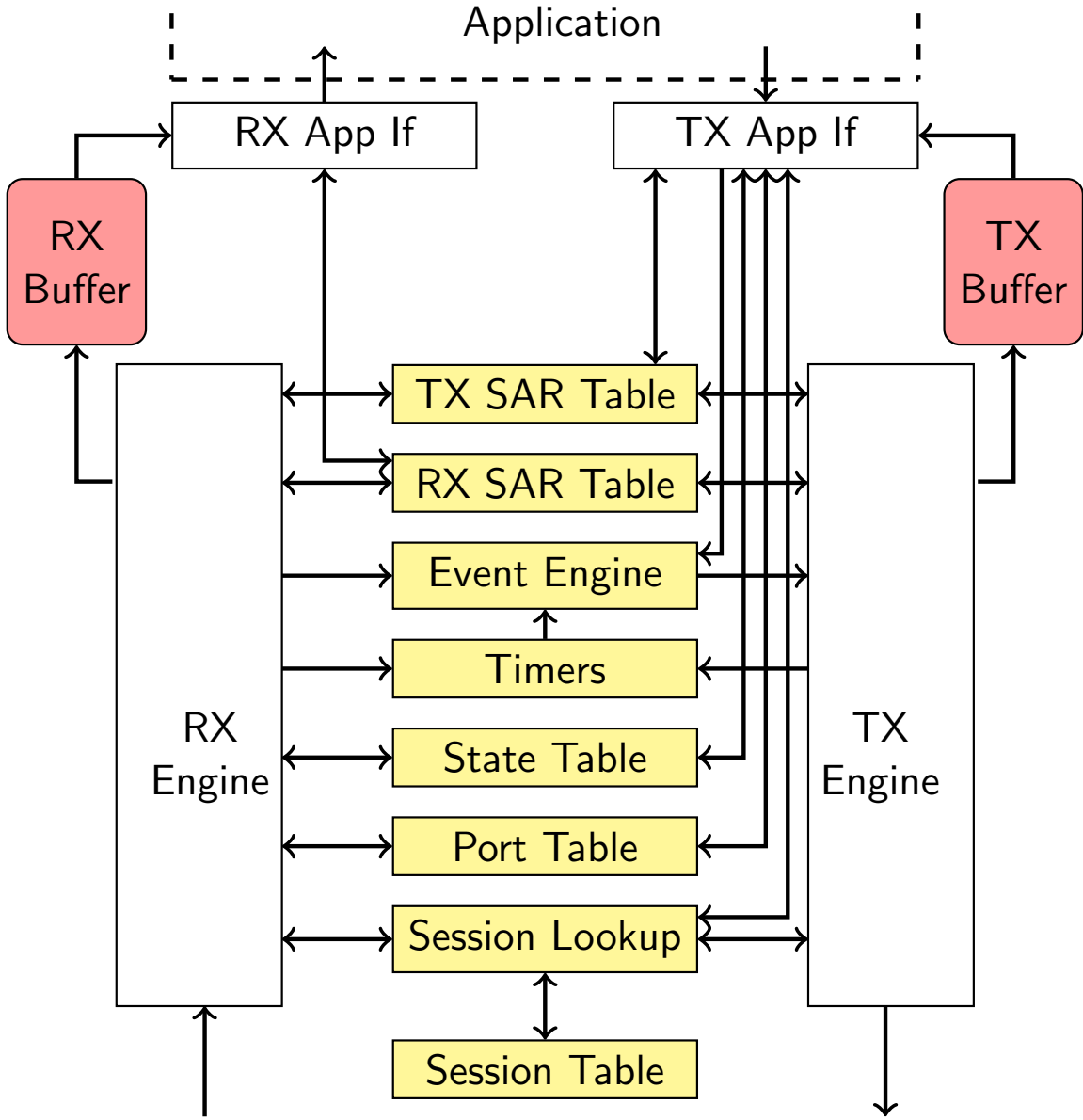
- Connection-oriented & stream-based protocol
  - Keep state for each connection
  - Data streams need to be segmented and assembled
- Acknowledged data transfer
  - Keep track of each segment
  - Data buffering is required for each transfer
- Various timers
  - Events/packets might be generated at any time
- Control flow
  - Slow-start, Congestion Avoidance, Delayed Acknowledgment

Stack Architecture



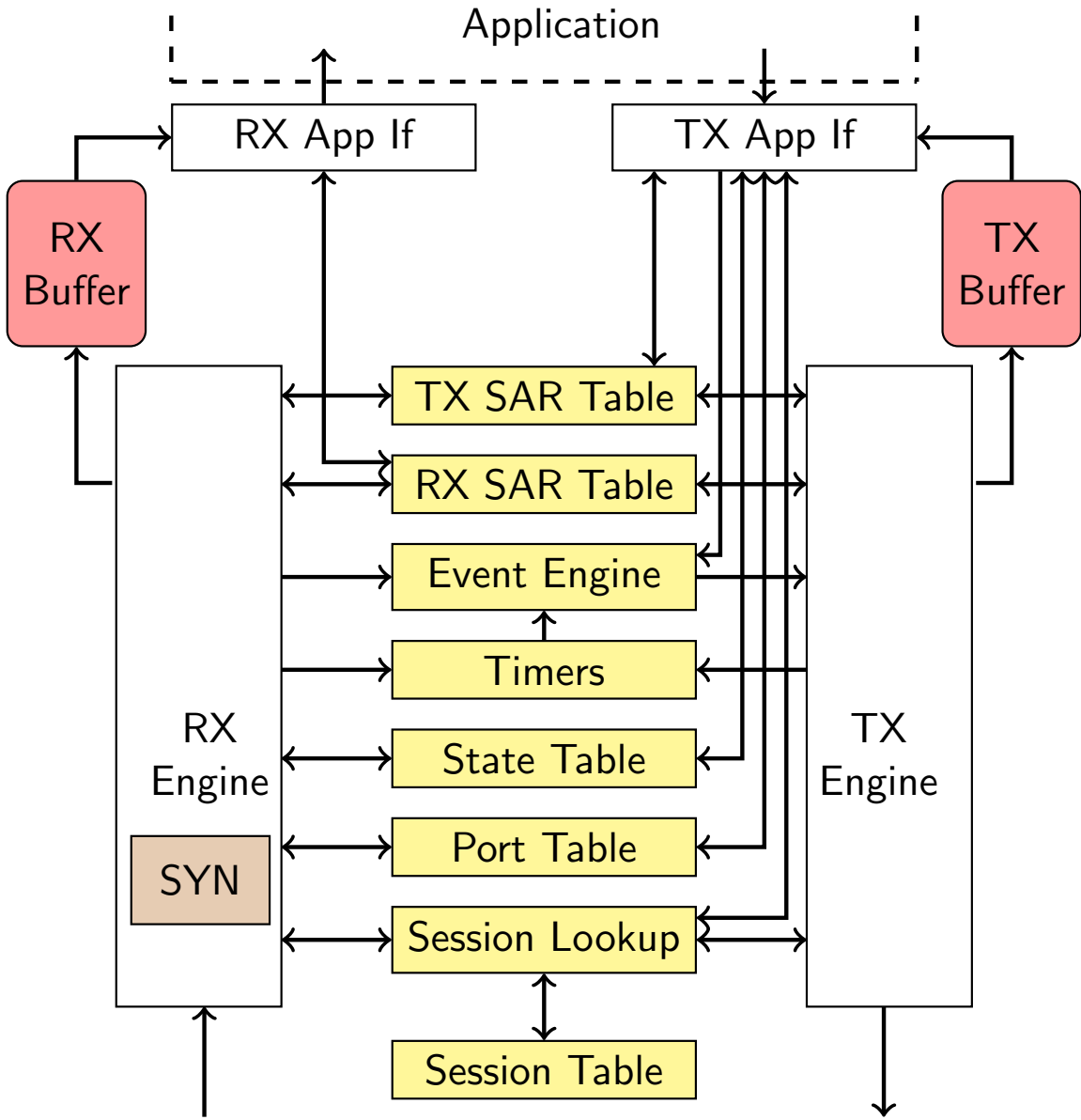
TCP Module Architecture

- Data-flow architecture
- Separation between data paths and state-keeping data structures
- Concurrent access to data structures in BRAM
- External buffers in main memory
- Scalable data structure



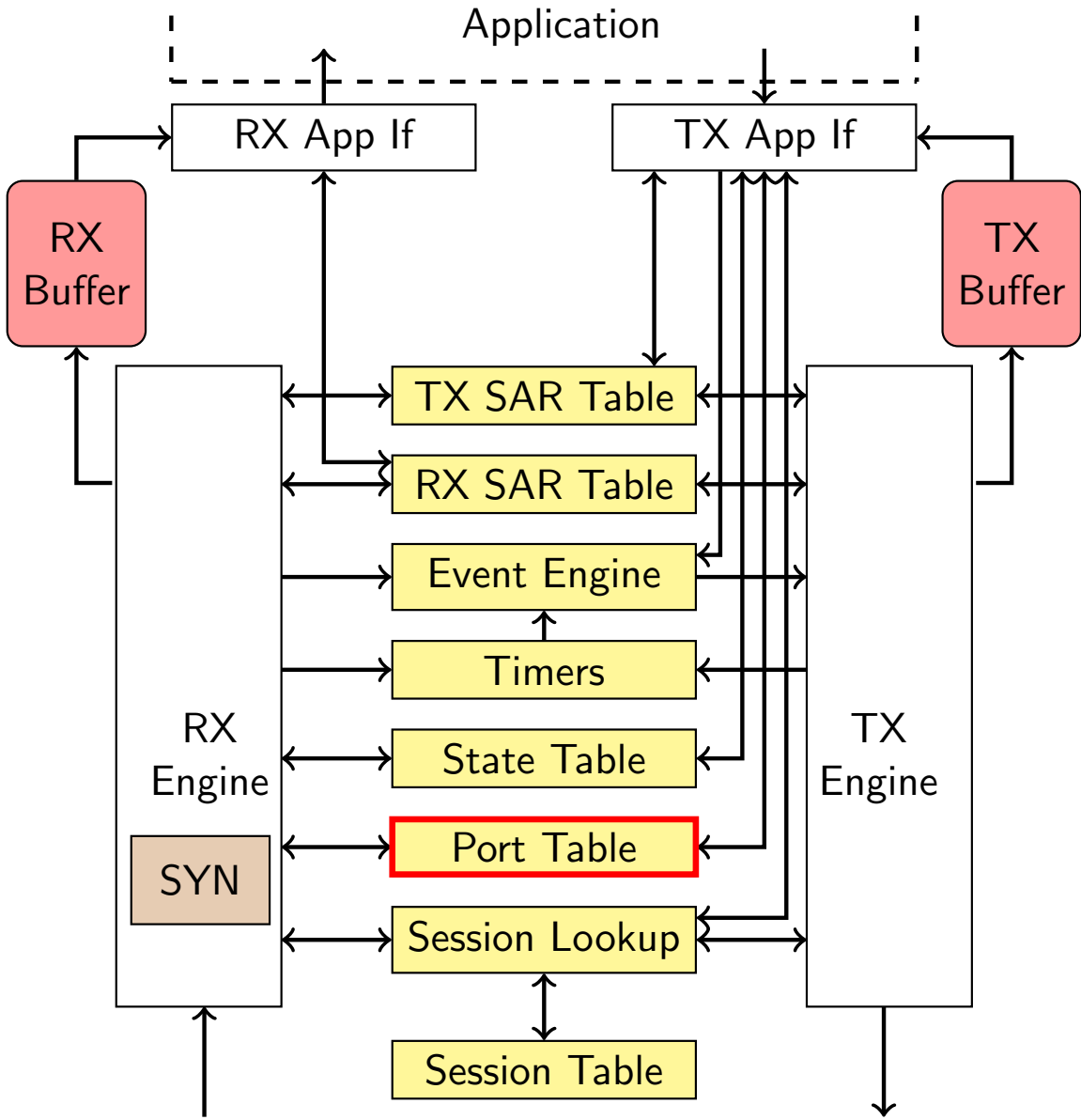
TCP Module - SYN Processing

1. SYN packet arrives



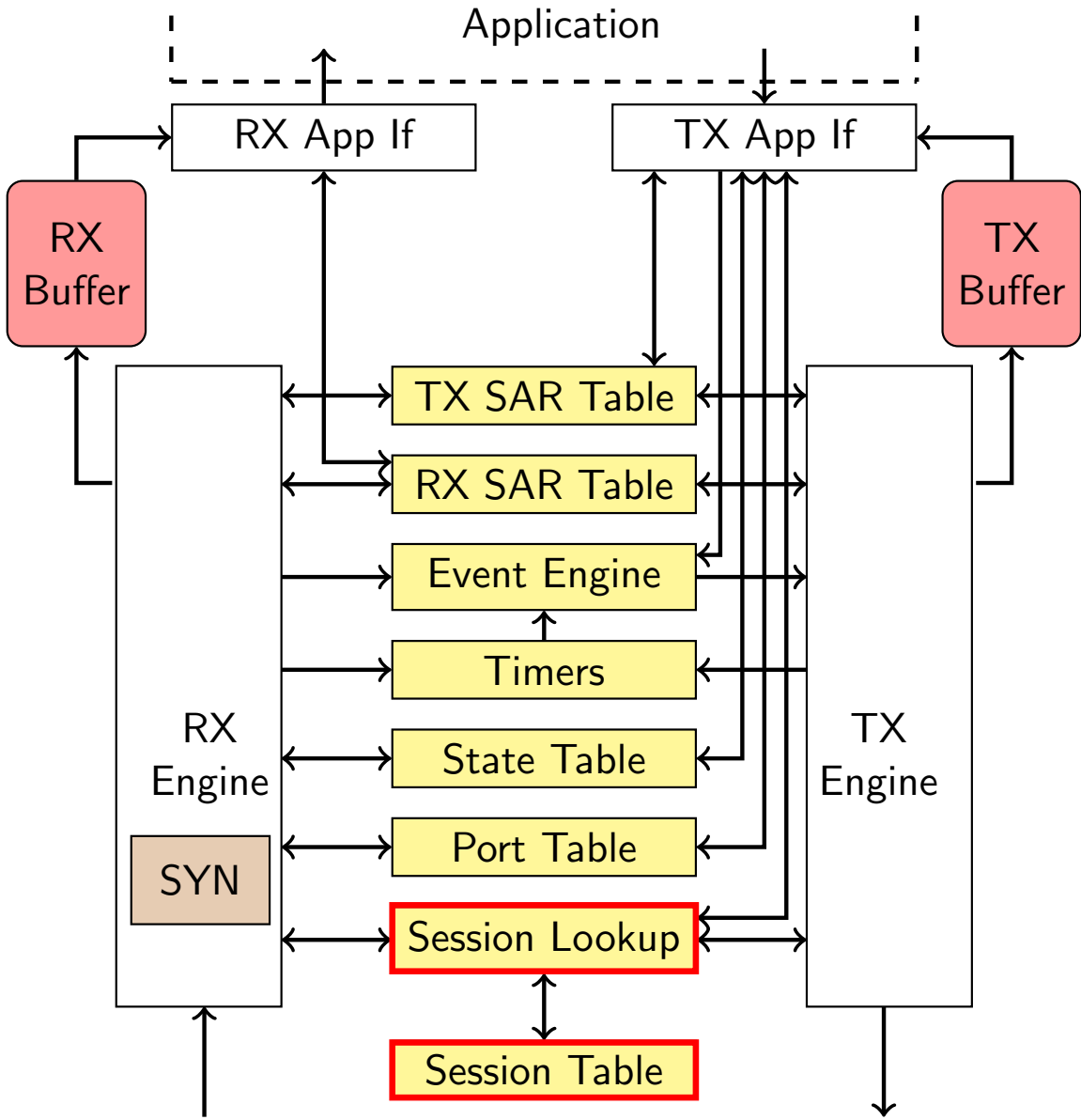
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open



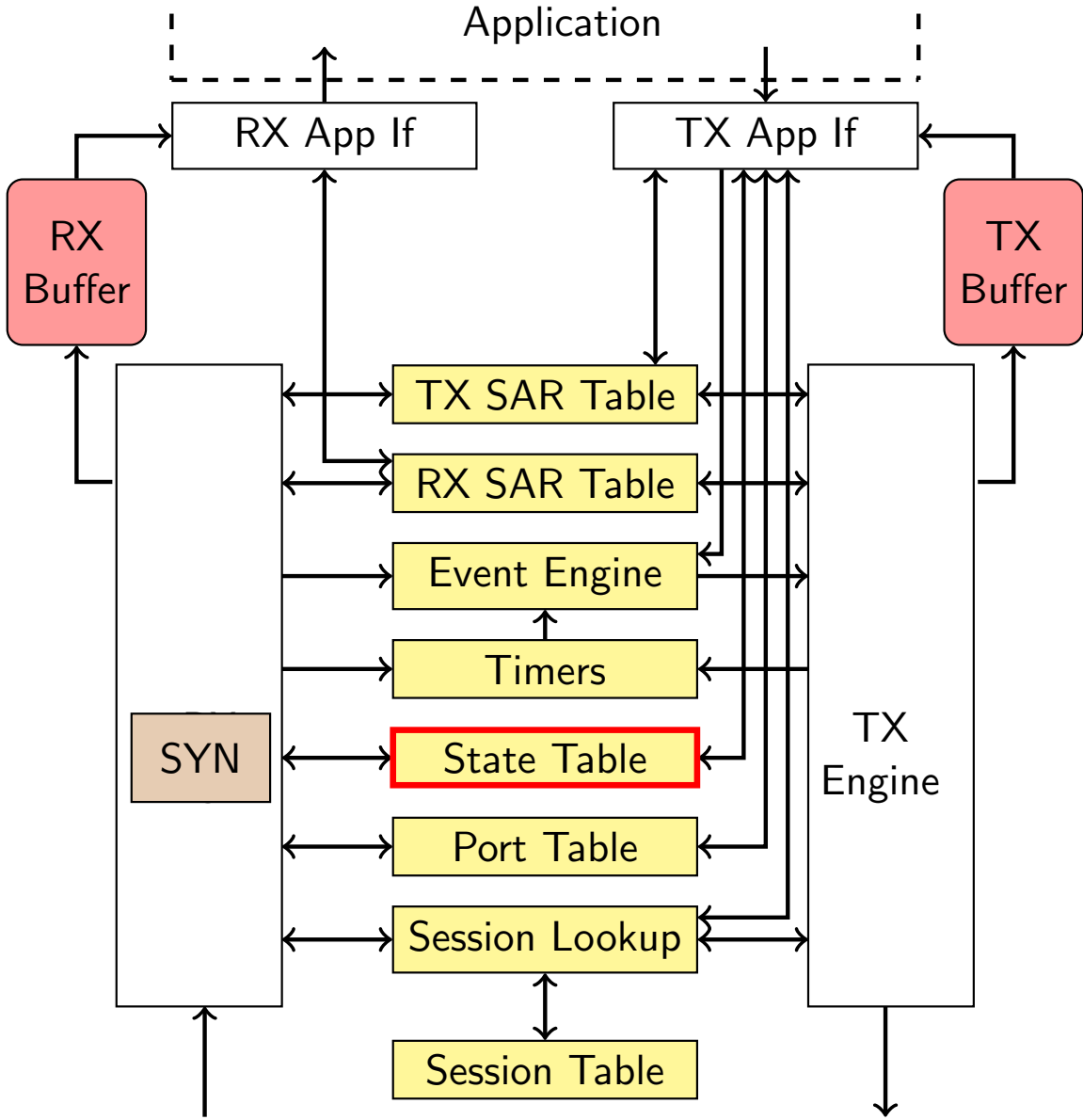
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID



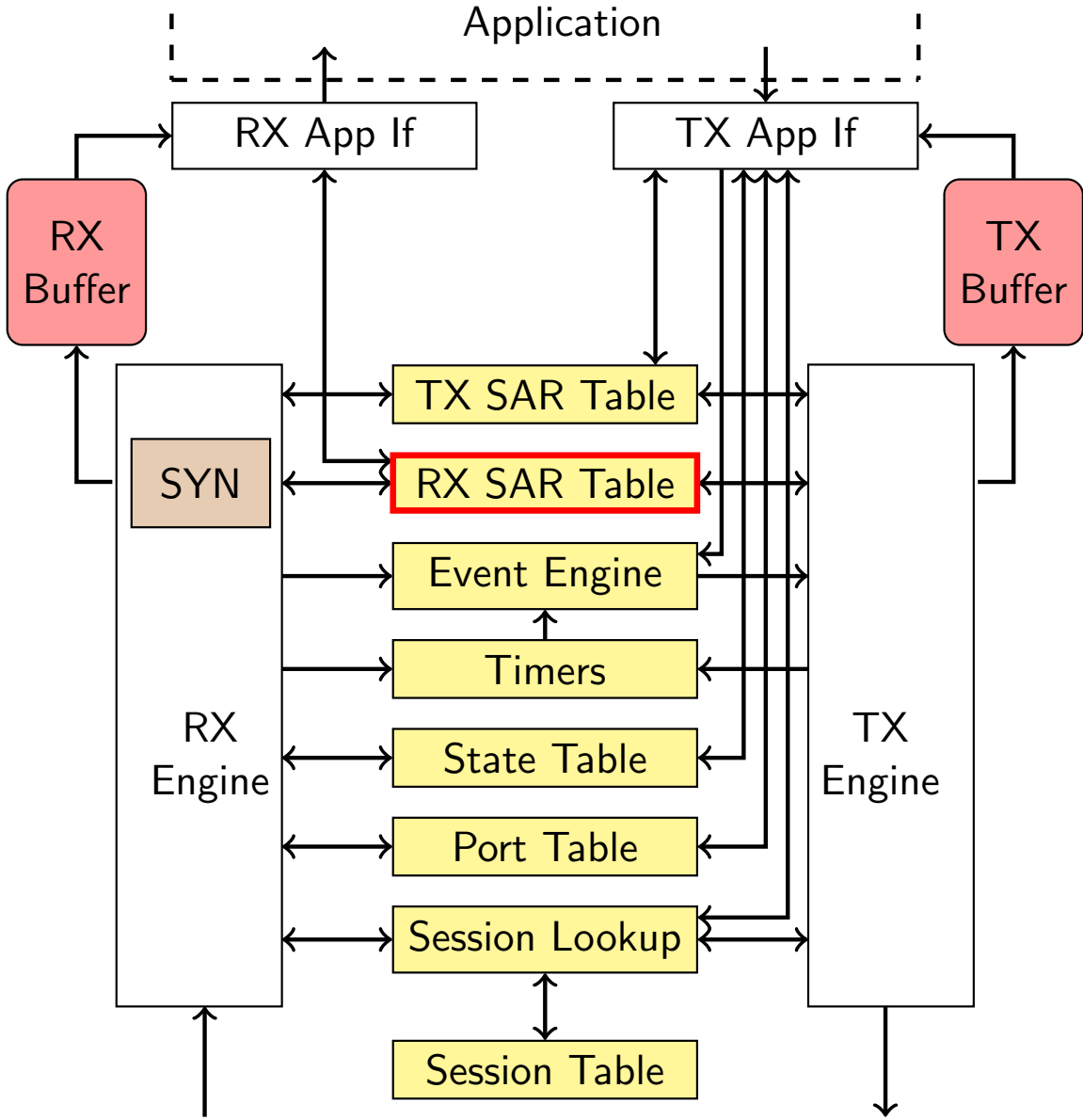
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD



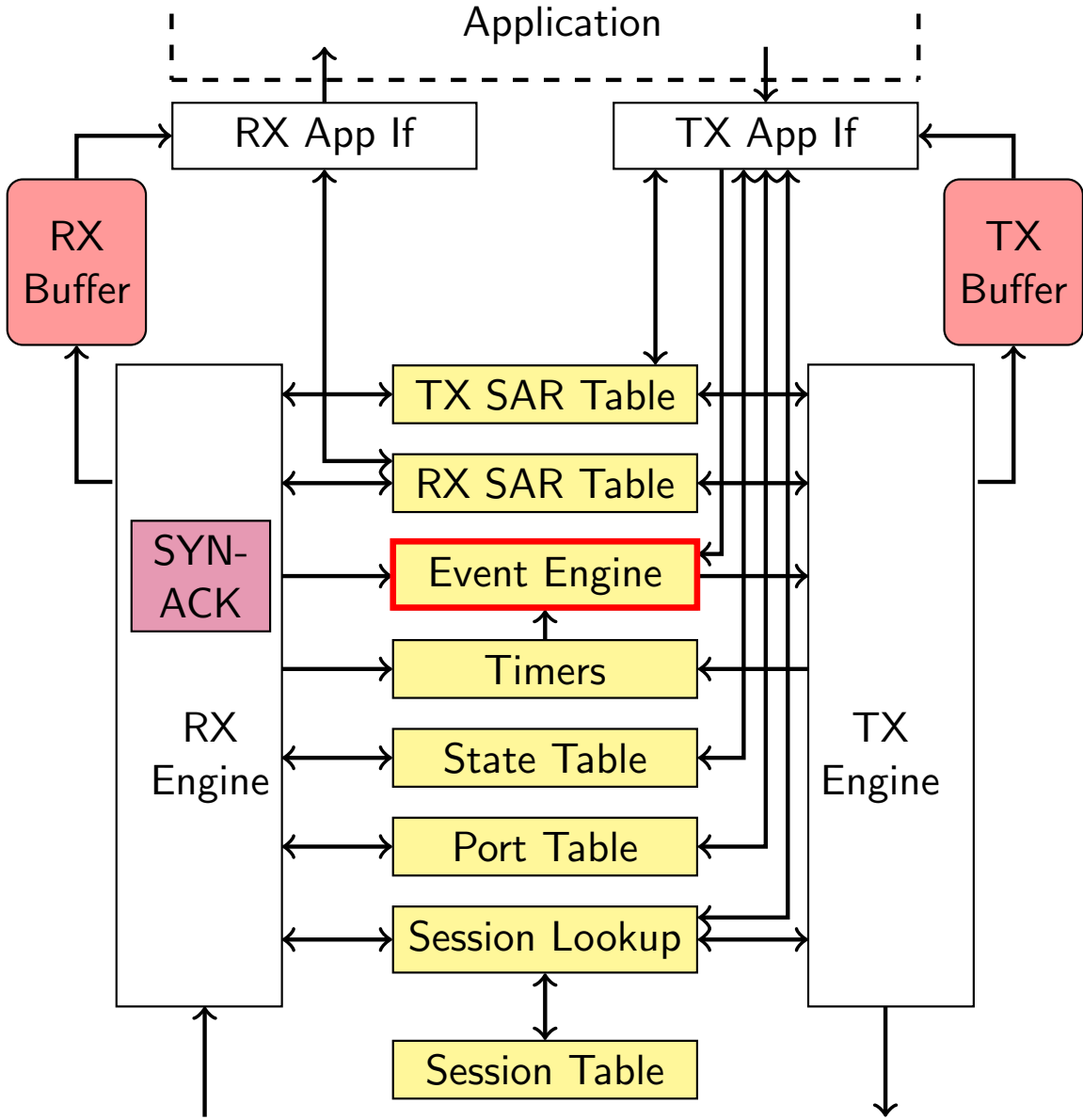
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table



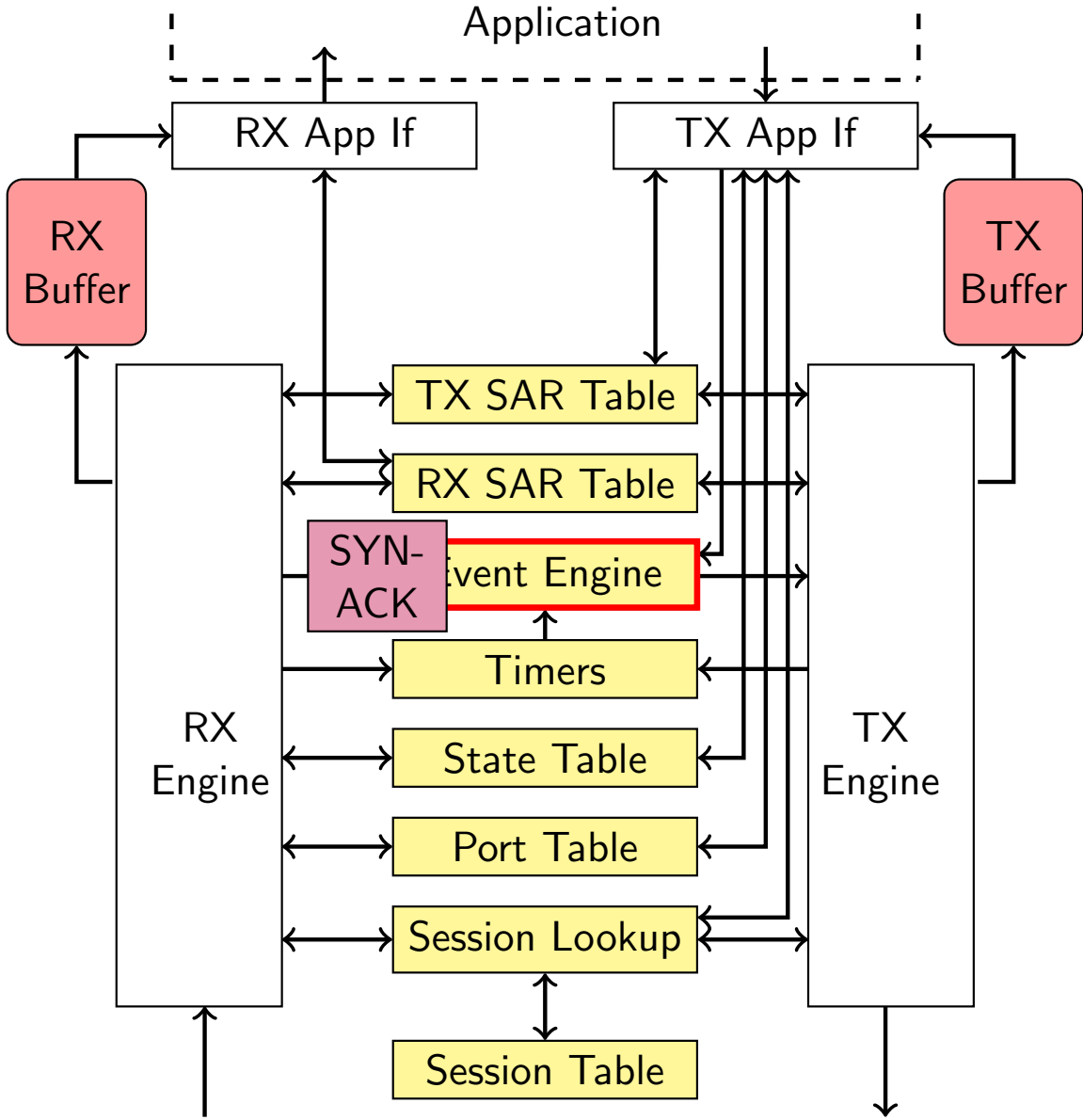
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table
6. Event is triggered



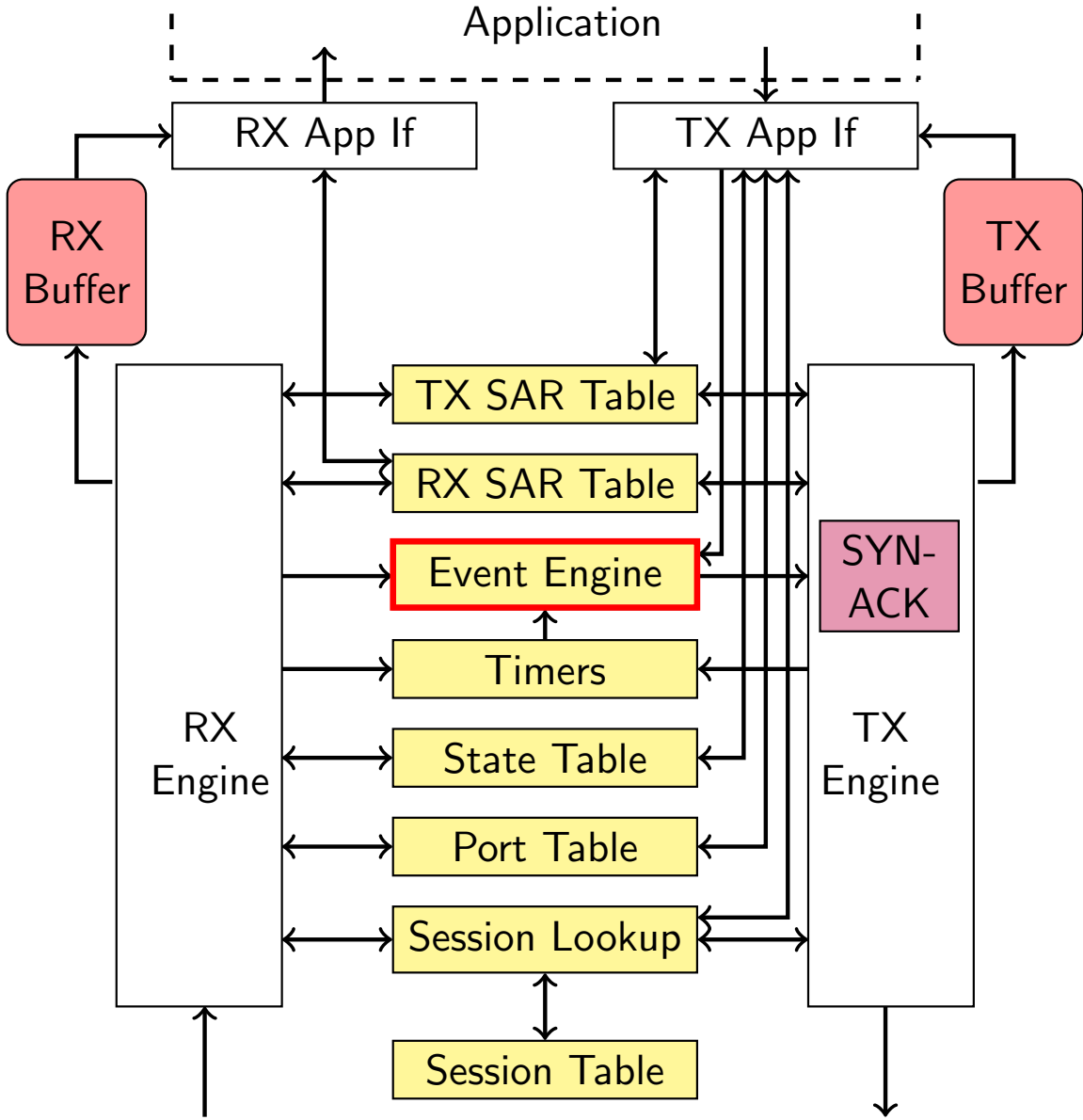
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table
6. Event is triggered



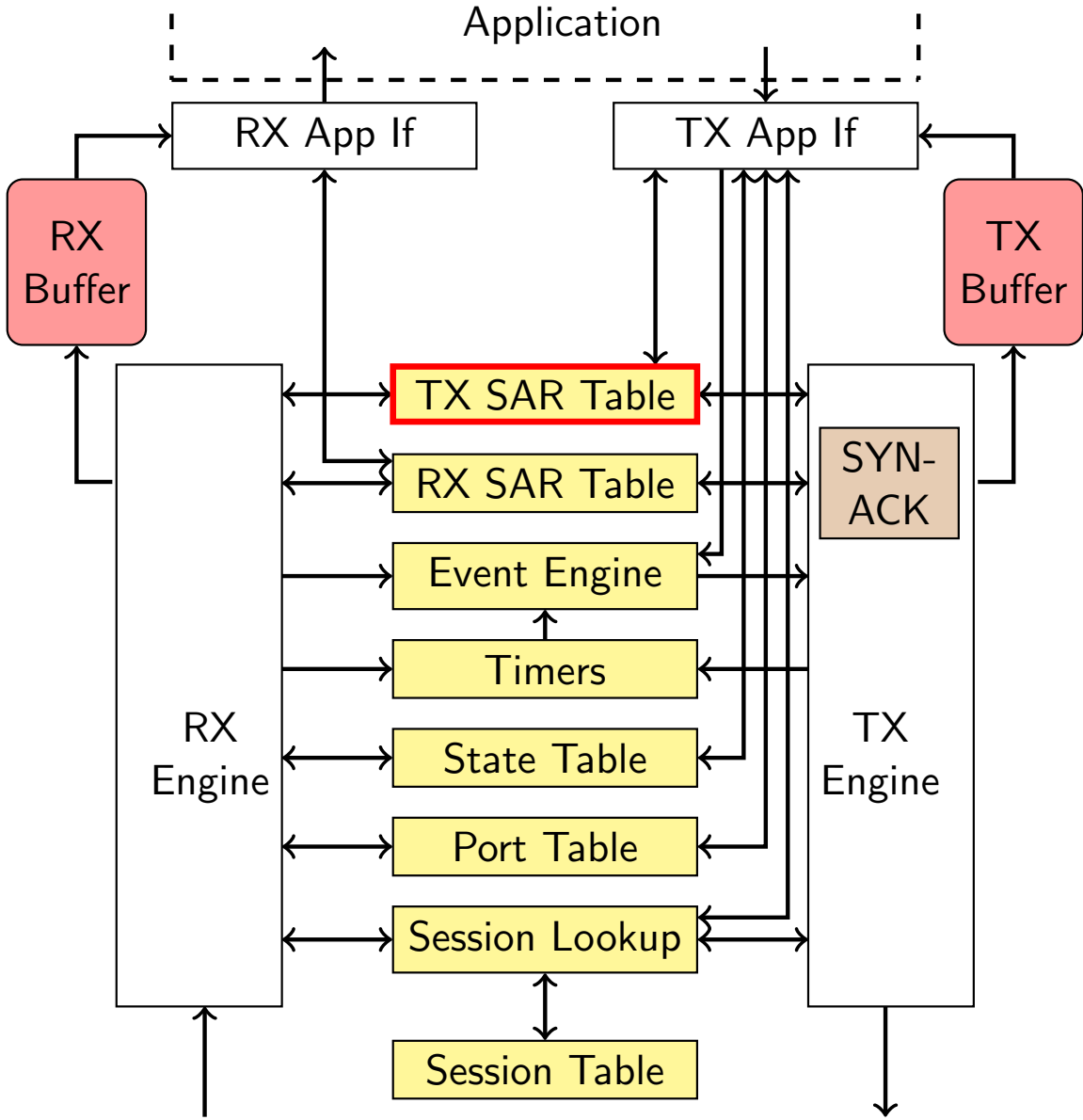
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table
6. Event is triggered



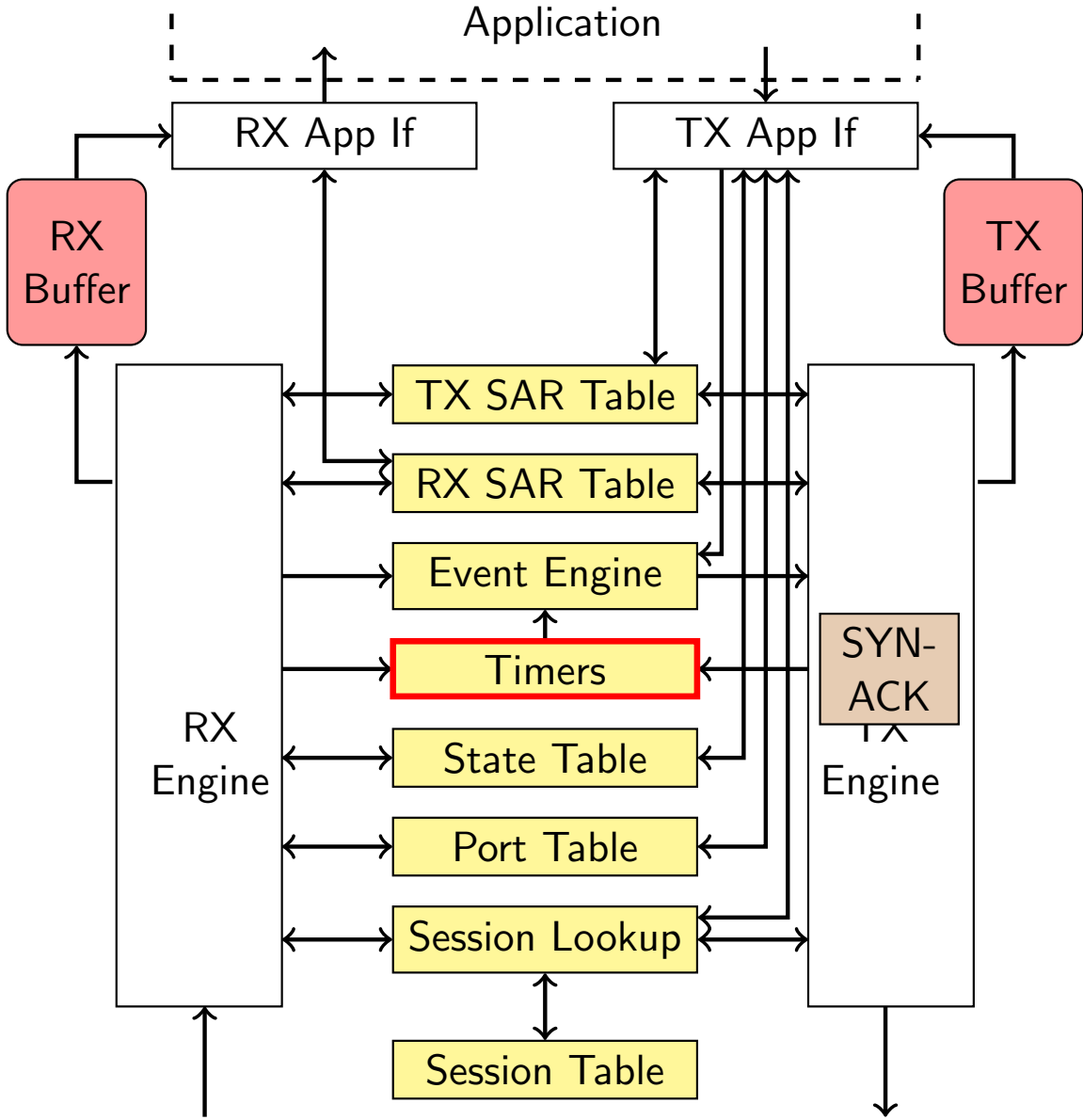
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table
6. Event is triggered
7. Initialize TX SAR Table



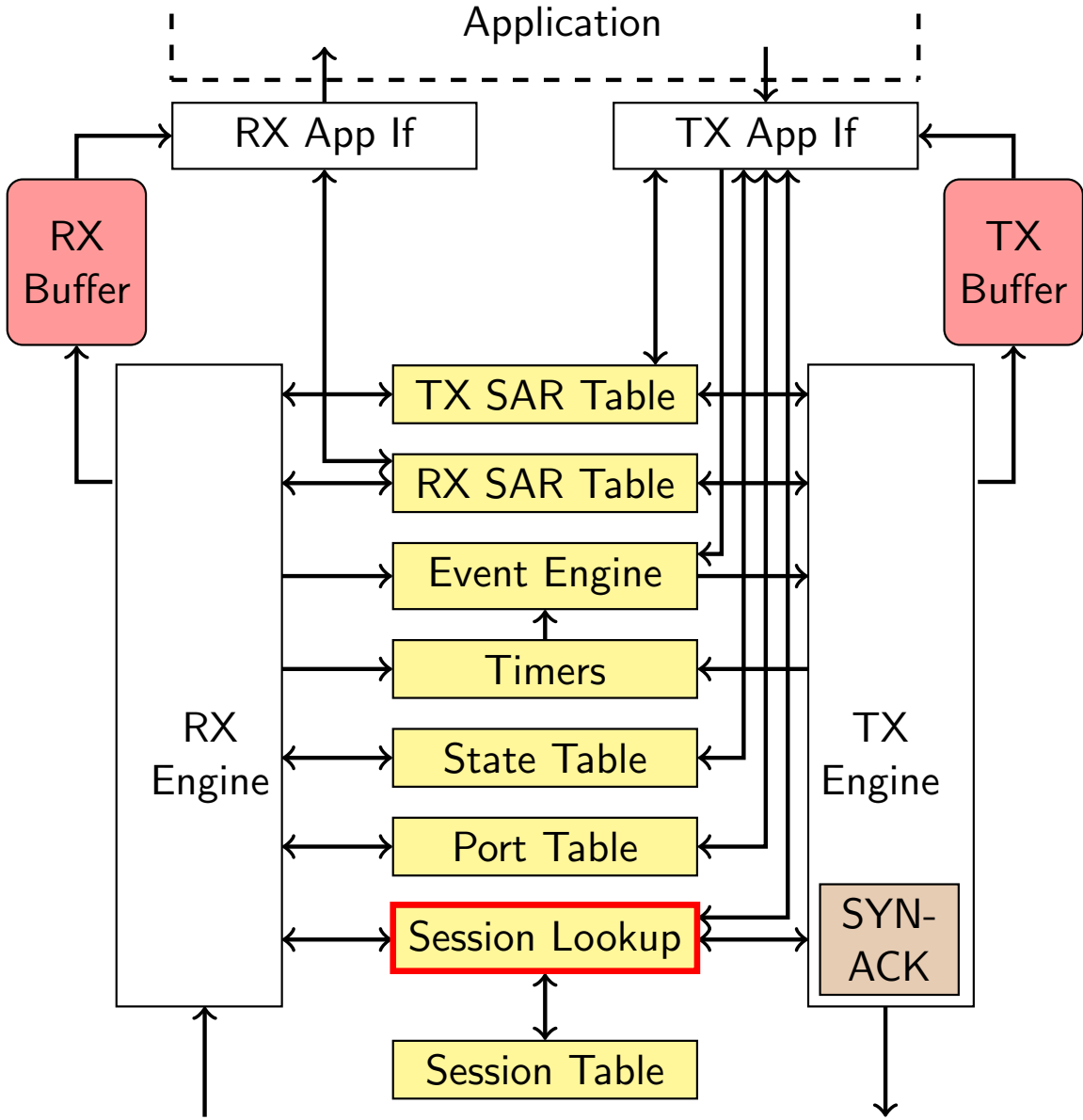
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table
6. Event is triggered
7. Initialize TX SAR Table
8. Set Retransmit-Timer



TCP Module - SYN Processing

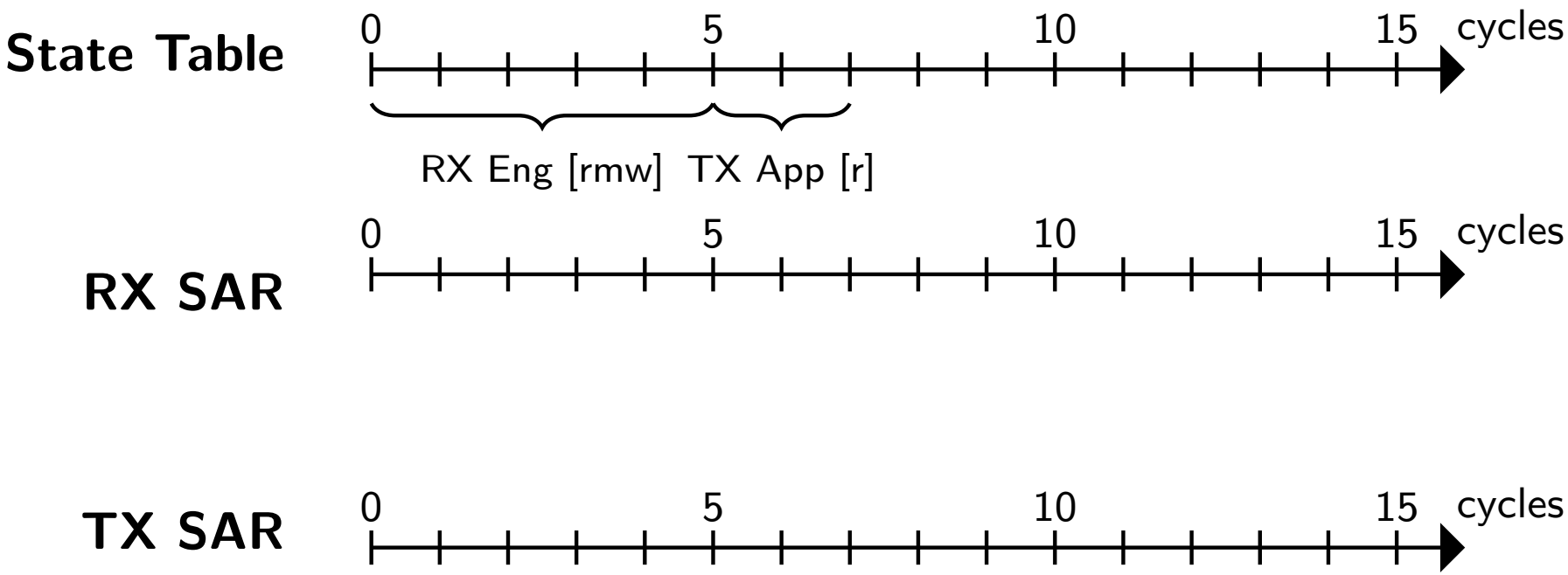
1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table
6. Event is triggered
7. Initialize TX SAR Table
8. Set Retransmit-Timer
9. Reverse session lookup



Data Structures Access Requirements

Minimum packet size is 84 bytes

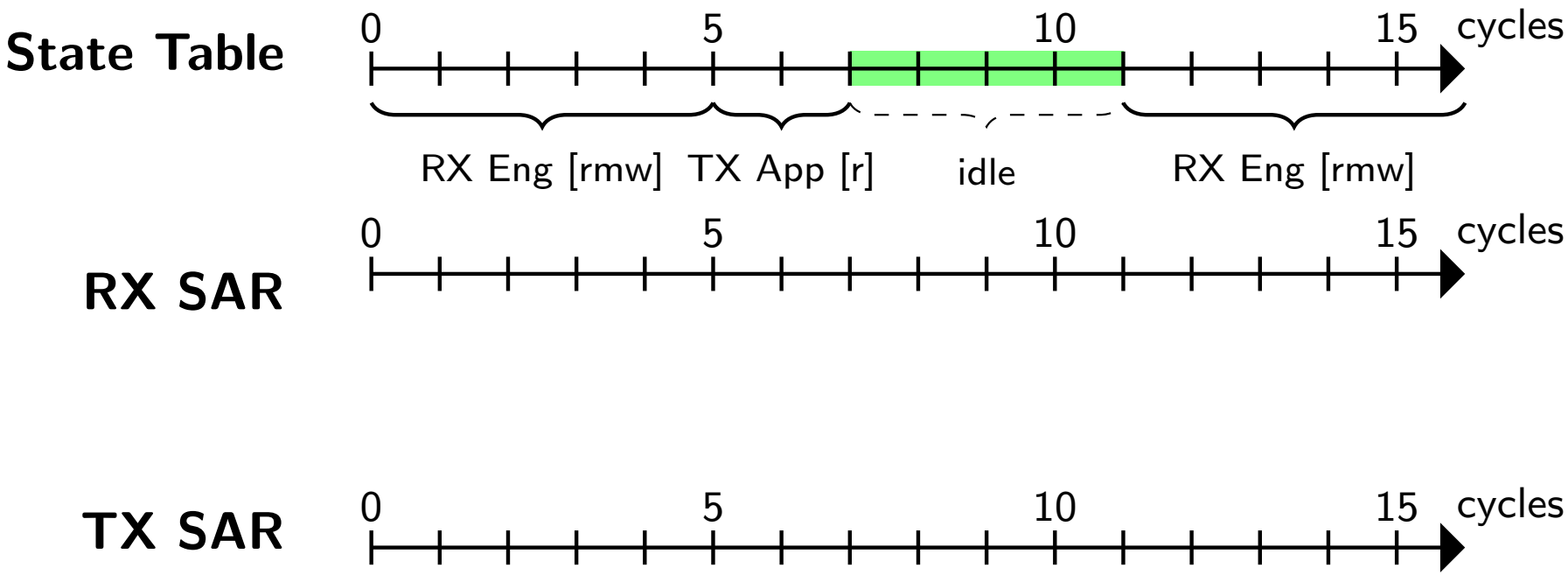
- For linkrate (8 B/c) processing, access time has to be within 11 cycles
- Data structures are shared between modules
  → The sum of all accesses (RX & TX path) can not exceed 11 cycles



Data Structures Access Requirements

Minimum packet size is 84 bytes

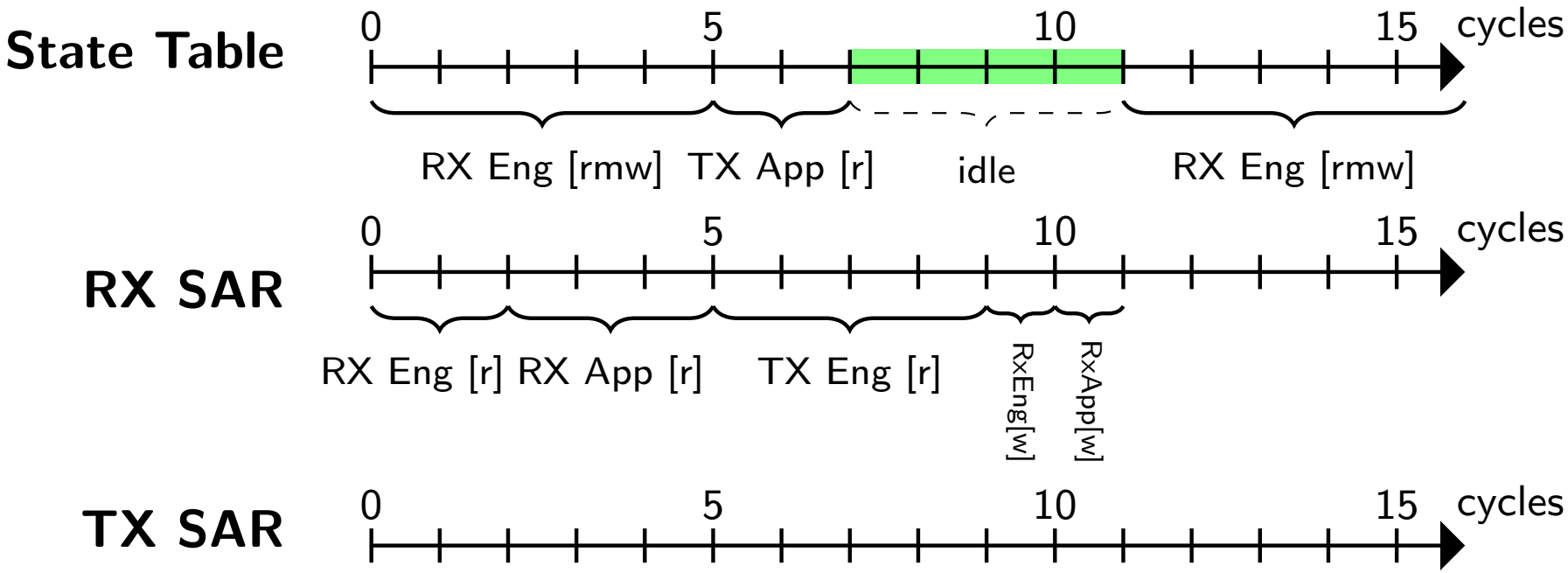
- For linkrate (8 B/c) processing, access time has to be within 11 cycles
- Data structures are shared between modules
  → The sum of all accesses (RX & TX path) can not exceed 11 cycles



Data Structures Access Requirements

Minimum packet size is 84 bytes

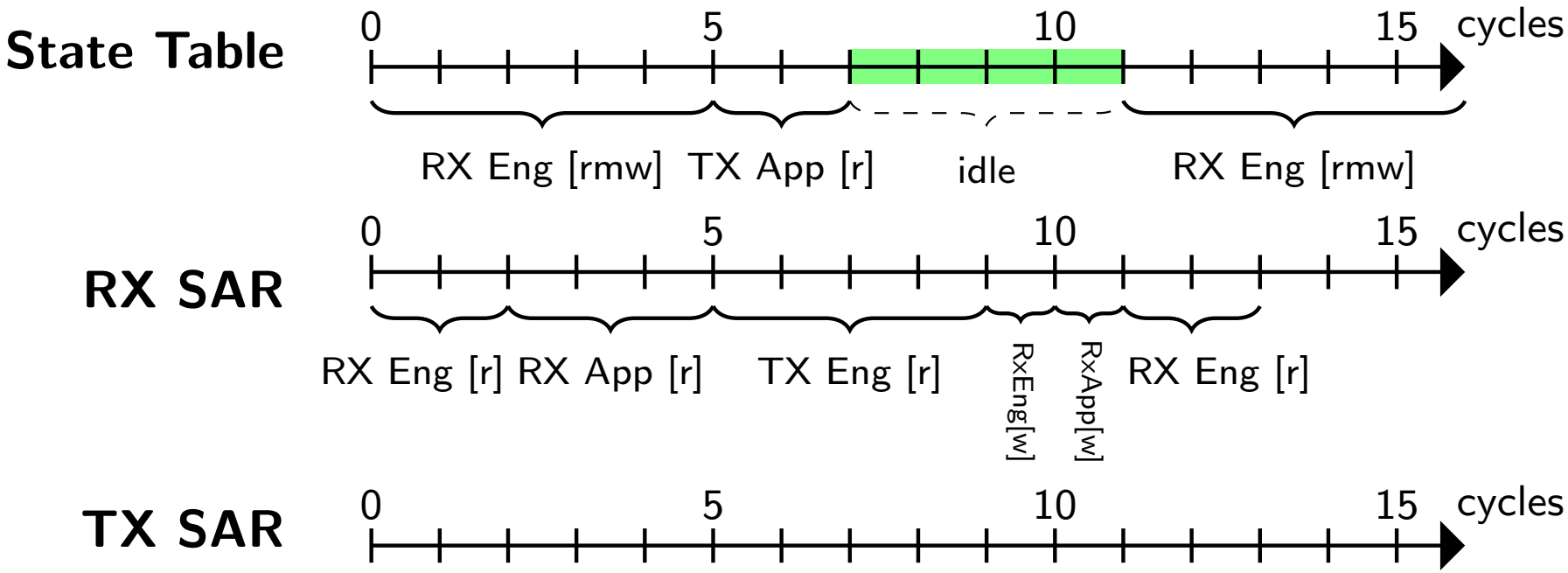
- For linkrate (8 B/c) processing, access time has to be within 11 cycles
- Data structures are shared between modules
  → The sum of all accesses (RX & TX path) can not exceed 11 cycles



Data Structures Access Requirements

Minimum packet size is 84 bytes

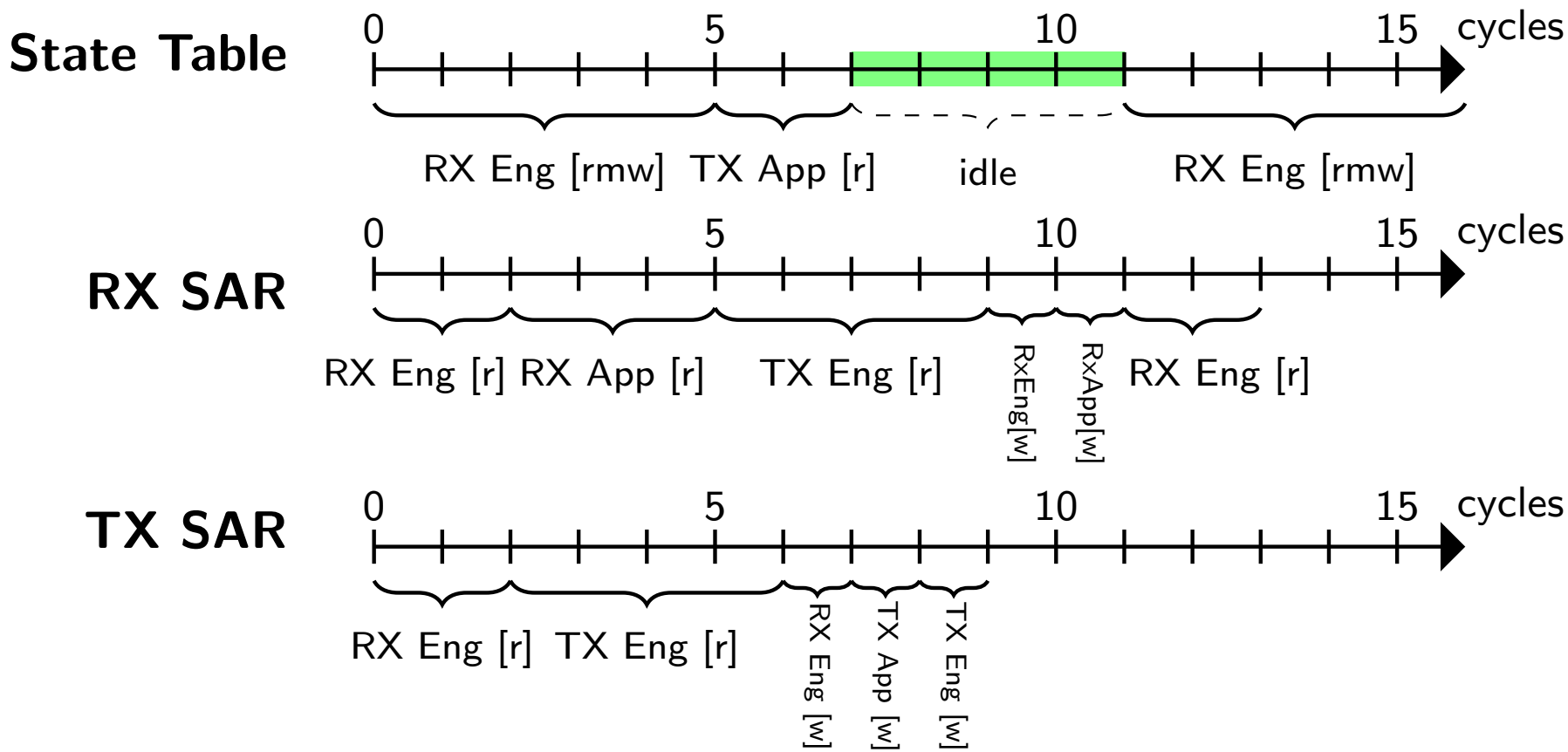
- For linkrate (8 B/c) processing, access time has to be within 11 cycles
- Data structures are shared between modules
  → The sum of all accesses (RX & TX path) can not exceed 11 cycles



Data Structures Access Requirements

Minimum packet size is 84 bytes

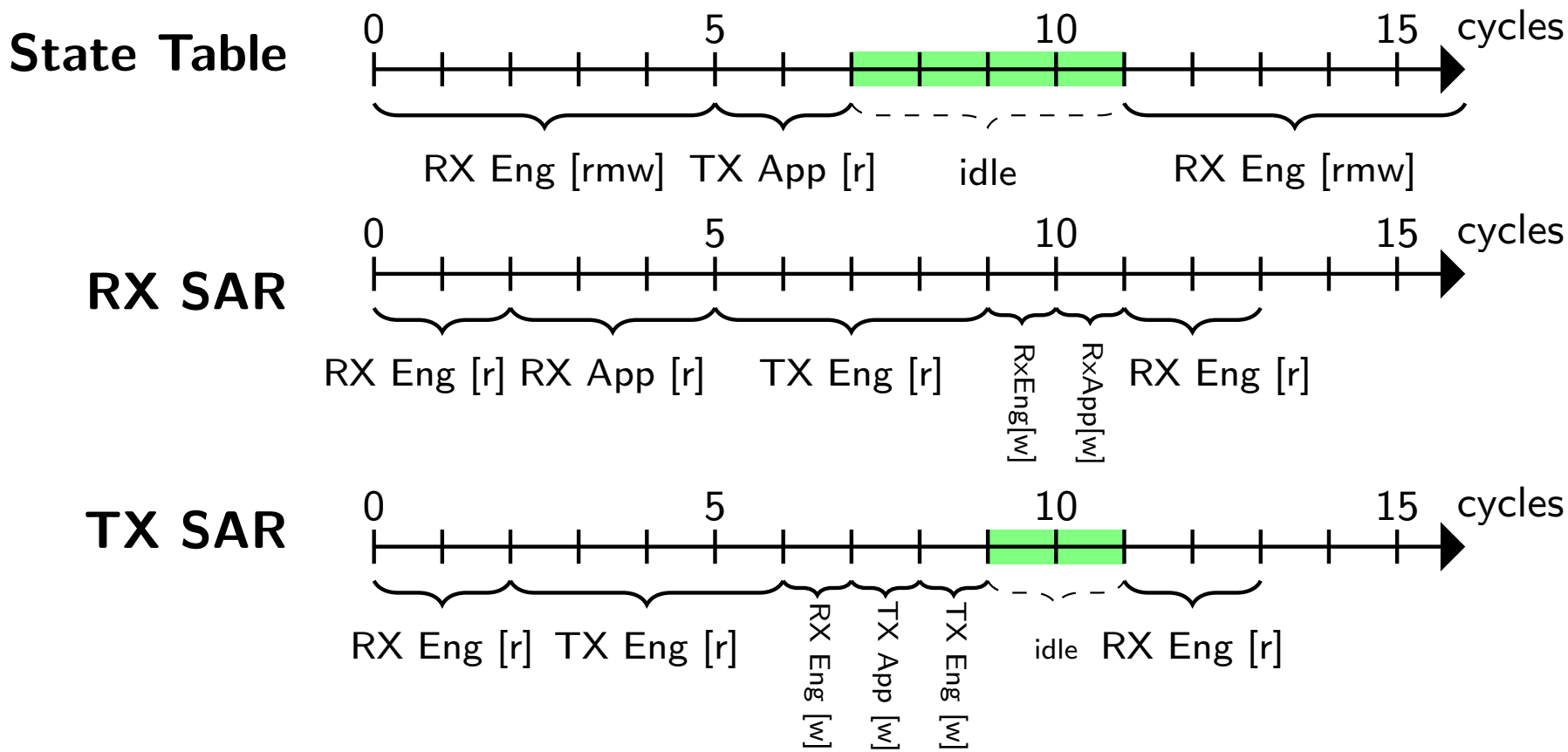
- For linkrate (8 B/c) processing, access time has to be within 11 cycles
- Data structures are shared between modules
  → The sum of all accesses (RX & TX path) can not exceed 11 cycles



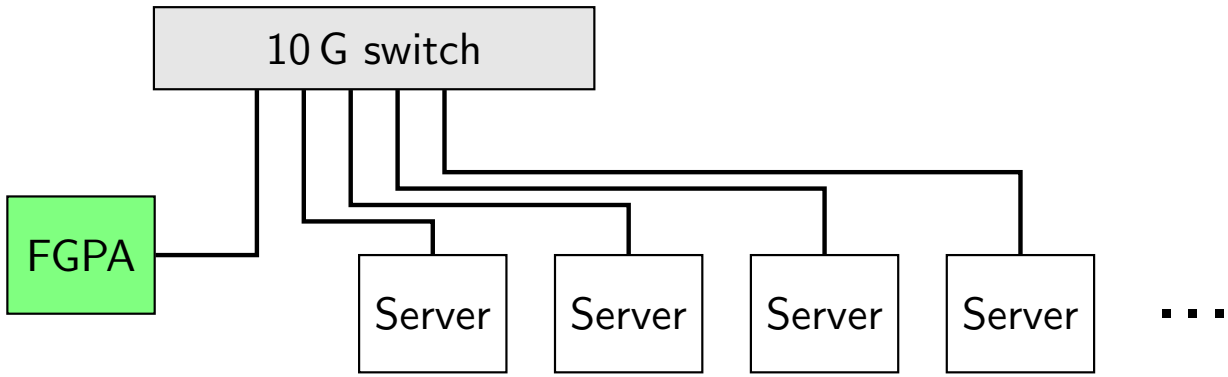
Data Structures Access Requirements

Minimum packet size is 84 bytes

- For linkrate (8 B/c) processing, access time has to be within 11 cycles
- Data structures are shared between modules
  → The sum of all accesses (RX & TX path) can not exceed 11 cycles

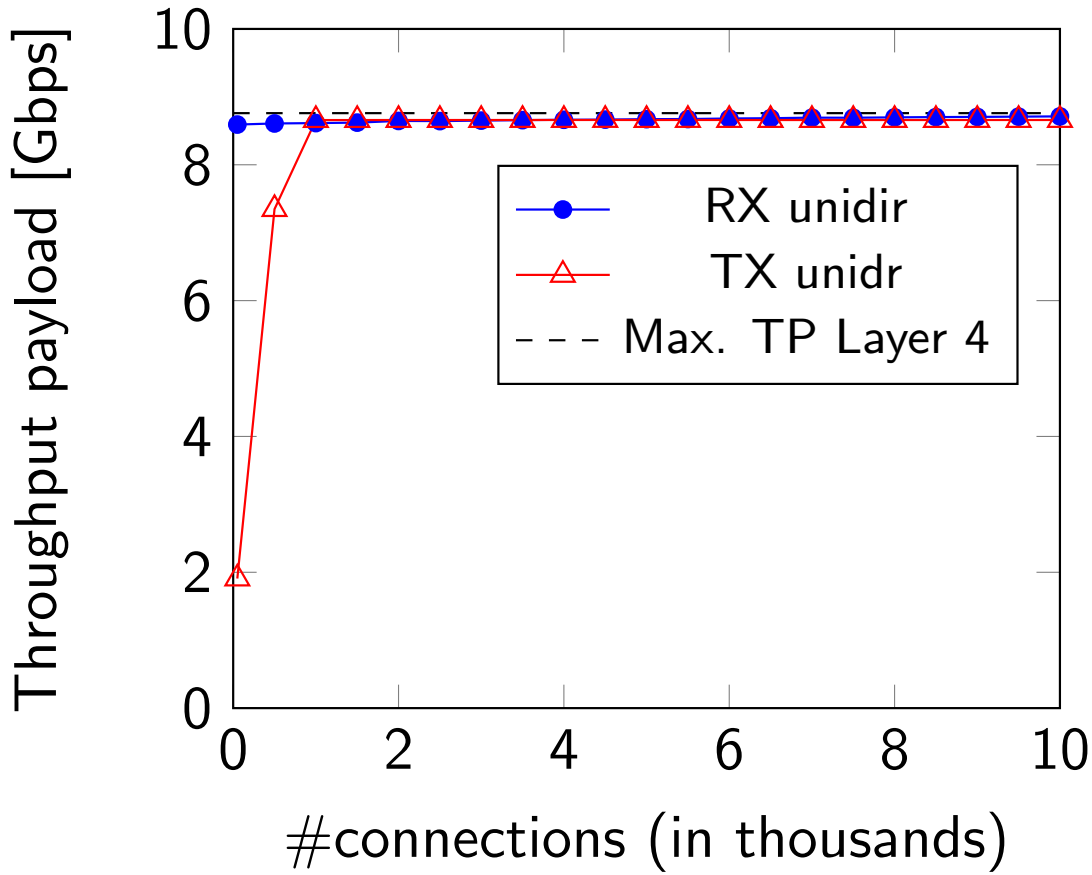


Evaluation - Setup



- TCP/IP stack running on VC709 evaluation board, Virtex7 XC7VX690T, 2x 4 GB DDR3, 10 G network interface
- 10 servers, 8-Core Intel Xeon E5-2609, 64 GB main memory, Intel 82599 10 G NIC, linux kernel 3.12
- Connected via a Cisco Nexus 5596UP switch

Evaluation - Performance



Maximum Segment Size (MSS) is 536 bytes. This leads to a theoretical maximum TCP throughput of 8.76 Gbps

Evaluation - Latency

Type	Path	Cycle [6.4 ns]	Time[μs]
SYN	SYN-ACK	176	1.1
Payload [1 B]	RX	170	1.1
	TX	131	0.8
Payload [536 B]	RX	375	2.4
	TX	402	2.6

Excluding PHY, MAC and application latency

Evaluation - Resources

	Network Interface	Memory Interface	TCP/IP Stack	**Total**	% of XC7VX690T Resources
FF	5,581	57,637	20,611	**83,829**	**9.6%**
LUT	5,321	43,591	19,026	**67,938**	**15.6%**
BRAM	8	36	279	**323**	**21.9%**

Conclusion

- Novel architecture for a TCP/IP stack
- Resource requirements scale linearly with number of concurrent connections
- Support for 10,000 concurrent connections
- Control flow features and out-of-order segment processing
- Reduced development time and increased design flexibility due to high-level synthesis

Future Work

- FPGA-based network interface could accelerate other functions such as compression, encryption
- Pushing data analytics and processing closer or into the network
- FPGAs as a microserver platform

Demo Tonight

- Key-value store on the FPGA using TCP/IP stack
- Serving thousands of clients concurrently
- Seamless integration with webserver running Apache and PHP