Analyzing Transport and MAC Layer in System-Level Performance Simulation

Subayal Khan, Jukka Saastamoinen,
Mikko Majanen, Jyrki Huusko
VTT Technical research Center of Finland,
FI-90570, Oulu, Finland
email:{subayal.khan,jukka.saastamoinen,
mikko.majanen,jyrki.huusko}@vtt.fi

Jari Nurmi
Tampere University of Technology,
Department of Computer Systems
P.O.Box 553 (Korkeakoulunkatu 1),
FIN-33101 Tampere, FINLAND
jari.nurmi@tut.fi

Abstract:
The modern mobile embedded devices support complex distributed applications via heterogeneous multi-core platforms. For the successful deployment of these applications, the scalability and performance analysis must be performed at all the layers of OSI model. This helps to identify the potential bottlenecks at different layers to perform the necessary optimizations. To achieve this goal, a framework is needed which accurately models the functionalities at different layers. The technical contributions described in this article include the extensions of ABstract inStruction wOrkLoad & execUtion plaTform based performance simulation (ABSOLUT) for the performance and scalability analysis of Transport and Medium Access Control (MAC) layers in the system level performance simulation. The article elaborates the design accuracy of the modeled components and their application in the context of M3 (multi-device, multi-vendor, multi-domain), which is a tri-layered conceptual interoperability architecture for embedded devices. These extensions pave the way towards the full coverage of the OSI model in the system-level performance simulation of distributed embedded systems. The network simulators for example ns-2, OMNeT++ and OPNET though provide detailed models of transport and MAC protocols but do not provide any framework such that these models can be used by the application workload models to mimic the real world use-cases. Also these models do not model the execution workload of these protocols on a particular execution platform and hence cannot be used in the architectural exploration of distributed embedded systems.

I. INTRODUCTION

Modern nomadic devices support computationally intense distributed applications by using heterogeneous multiprocessor architecture platforms. Deployment of a new distributed application is challenging not only due to the heterogeneous parallelism in the platforms [1], but also due to performance and energy constraints. Furthermore, these devices employ diverse communication, transport technologies and application-level protocols to enable information sharing and synchronization among the processes of distributed applications. These technologies enable complex use-cases spanning multiple devices. To evaluate the feasibility of these use-cases, the system-level performance simulation methodology must identify the potential bottlenecks at each layer of the OSI model so that the appropriate optimizations can be performed. The performance analysis of protocols spanning different layers of OSI model require a framework which models these protocols with reasonable accuracy while maintaining a good simulation speed.

In case of non-distributed applications, the processes use inter-process communication (IPC) for synchronization of tasks and the transport, Datalink and Physical layers of the OSI-Model do not play any role. The system-level performance simulation of non-distributed application requires application workload models, platform capacity models and workload models for external libraries. ABSOLUT performance simulation methodology has been already used to evaluate such use-cases [2] and [3].

In the case of distributed applications, the use-cases span multiple devices and the processes communicate with each other via a transport API. The transport layer APIs are implemented on top of Layer 2 of the OSI model which in turn makes use of physical layer. Therefore apart from the performance evaluation of the platform components, the performance evaluation of the applied transport technology, MAC protocol and transmission techniques must be performed. Afterwards, the performance of same application level workload models can be evaluated with other available alternatives of transport, MAC and transmission technologies. The design space is thus much bigger and spans all layers of the OSI model.

The main contribution of this article is to describe the extensions made to ABSOLUT for the performance evaluation of distributed applications. A detailed survey of performance simulation techniques has been presented in [4], therefore the landmark performance simulation methodologies are not discussed in this article.

Rest of paper is organized as follows: Section 2 briefly explains ABSOLUT methodology. Section 4 describes the extensions made to ABSOLUT for enabling the analysis of Layer-2 in System-level performance evaluation. We first elaborate a general method for developing operating system (OS) services for Transport, MAC and physical layers of OSI models. These extensions pave the way towards the full coverage of the OSI model in the system-level performance simulation. These services are implemented by using freely available tools and libraries such as SystemC [5] and itpp library [6]. Afterwards we focus on the modeling and integration of IEEE 802.11 MAC and transport layer of OSI model to ABSOLUT. In section 5, the modeled components are used for the MAC and transport layer scalability and performance analysis of M3 [7]. The obtained simulation results are compared with the ns-2 [8] and OMNeT++ [9] network simulators under different simulation scenarios. Conclusions and Future work are mentioned in Section 6.

II. ABSOLUT

ABSOLUT follows the Y-chart model consisting of application workload and platform model [10]. The workload models are mapped to the platform for transaction-level performance simulation in SystemC [2].

3.1 Application Workload Model

The workloads consist of four layers .i.e., main workload, application workload, process workload and function workload as shown in Figure 1.

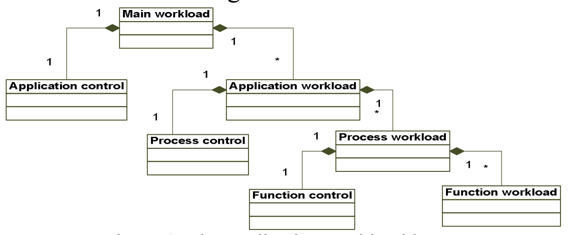


Figure 1: The application workload layers

3.2 Execution Platform Model

The platform model is an abstract hierarchical representation of actual platform architecture. It is composed of three layers: component layer, subsystem layer, and platform architecture layer as shown in Figure 2. Each layer has its own services, which are abstract views of the architecture models. Services in subsystem and platform architecture layers are invoked by application workload models.

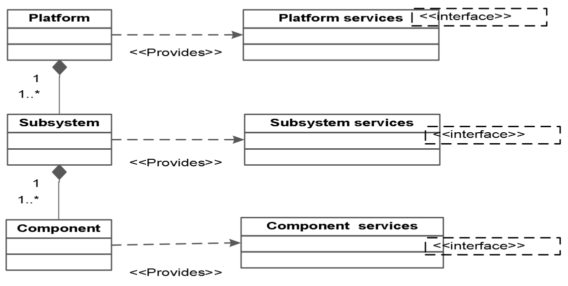


Figure 2: The platform architecture model layers.

III. EXTENDING ABSOLUT FOR ANALYSIS OF LAYER-2

The performance modelling of distributed applications via ABSOLUT demands the modelling and integration of Transport and MAC protocols, modulation techniques, coding schemes and channel models. We now elaborate the modelling and integration of these components to ABSOLUT.

3.1 Design and integration of OS Services

Extension of ABSOLUT for analysing the performance of protocols operating at different layers of OSI model during system-level performance simulation for architectural exploration demands a mechanism for instantiating new H.W and S.W services. These services are registered to the operating system and are used by the application workload models. Furthermore the services operating at a higher layer for example transport-level services (such as TCP) use Data-link level services such as IEEE 802.11 MAC protocols for the transmissions of frames of a packet. These services are created by deriving them from the *OS_Service* base class as shown in Figure 3 which implements the *Generic_Serv_IF*.

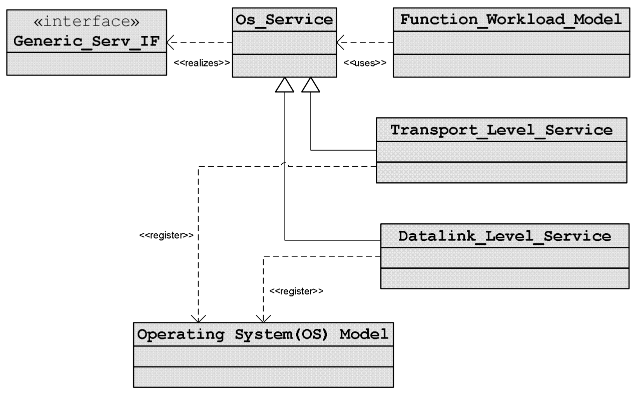


Figure 3: OS services implemented to model use cases spanning multiple devices and for modeling BSD API as OS services.

Implementation of *Generic_Serv_IF* by the *OS_Service* base enables the process-level application workload models or higher level services request a certain service by its name and invoke the functionality implemented by the derived service. This is shown in Figure 4.

```
SID=Use_Service(”Serv_Name”);
Wait_Service(SID);
```

Figure 4: Accessing an OS_Service via Generic_Serv_IF

3.2 Accessibility and Hierarchy of *OS_Services*

The *OS_Service* base class implements the functionality related to scheduling of service requests of processes via priority queues and informs the requesting process on service completion after taking it to running state again. This is shown in Figure 5.

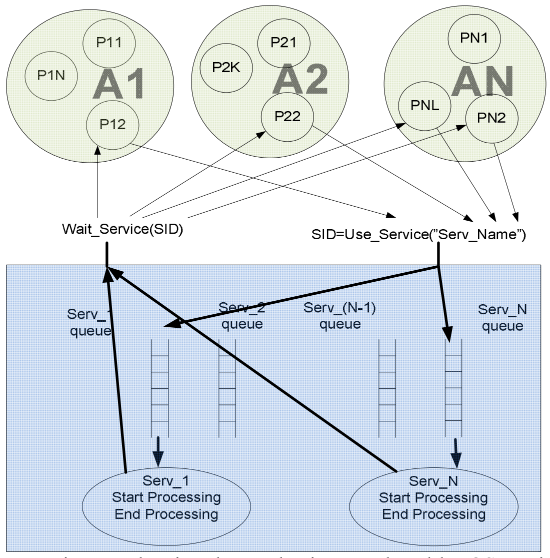


Figure 5: Diagram showing the mechanism employed by OS services to execute requests of processes.

The derived services implement the service-specific functionality making service modelling straight forward. The services at upper layers make use of services at lower layer .e.g., Transport layer use Data-Link layer services.

The services are accessed from the platform using the service name assigned while registration to the OS model. For example BSD socket API function “send()” can be modelled as an *OS_Service* and registered by a unique service name for example “PktTx” to the OS. It can then accessed by the process workload models by using its unique service name via *Generic_Serv_IF*. This is shown in Figure 4 and Figure 6.

```
//The processor model
ARM_Crtx_Proc_ptr=new Scalable_MultiCore_CPU
"m_ARMCortex_A9_MP_Processor");

//Creating the operating system (OS) model
m_os = new Generic_serv_op_sys
("os",ARM_Crtx_Proc_ptr->GetProcessorCores(),
m_os_addr);

//Send() API function registered as “PacketTx” to OS
Pkt_Tx_Service= new Packet_Tx_Serv("Transmit_Packet",m_os);
Serv_type Msg_serv_type = { SERV_TYPE_LOCAL };
m_os->register_service(Pkt_Tx_Service,"PacketTx",
Msg_serv_type);

//This service handles transmission of single frame via
//IEEE 802.11 DCF. Registered by name “FrameTx” to OS
Frame_Tx_Service= new Frame_Tx_Serv("Transmit_Frame",m_os);
Serv_type Frame_serv_type = { SERV_TYPE_LOCAL };
m_os->register_service(Frame_Tx_Service,"FrameTx",
Frame_serv_type);
```

Figure 6: Registration of services to the operating system (OS) model

3.3 Implementation and integration of MAC and Transport Level OS_Services

IEEE 802.11 distributed coordination function (DCF) requires a station wanting to transmit, to first listen to the channel to check its status (occupied or not) for a DCF Interframe Space (DIFS) interval. If the channel is found busy during the DIFS interval, the station defers its transmission. In a network where a number of stations contend for the wireless medium, if multiple stations sense the channel busy and defer their access, they will also virtually simultaneously find that the channel is released and then try to seize the channel. As a result, collisions may occur. In order to avoid such collisions, DCF also specifies random back off, which forces a station to defer its access to the channel for an extra period. DCF also has an optional virtual carrier sense mechanism that exchanges short Request-to-send (RTS) and Clear-to-send (CTS) frames between source and destination stations during the intervals between the data frame transmissions. The IEEE 802.11 DCF can be shown in the form of a flow chart as in Figure 7.

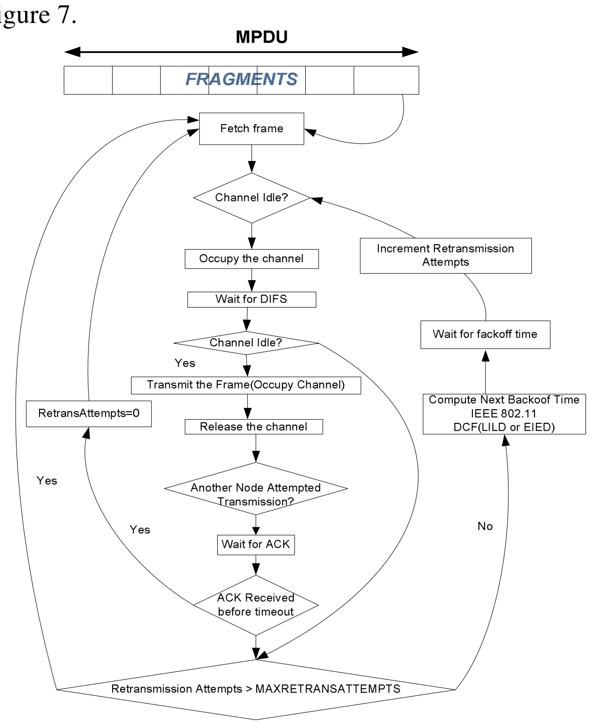

MPDU
FRAGMENTS
Fetch frame
Channel Idle?
Occupy the channel
Increment Retransmission Attempts
Wait for DIFS
Wait for fackoff time
Channel Idle?
Yes
Transmit the Frame(Occupy Channel)
Compute Next Backoof Time IEEE 802.11 DCF(LILD or EIED)
RetransAttempts=0
Release the channel
Yes
Another Node Attempted Transmission?
Yes
No
Wait for ACK
ACK Received before timeout
Retransmission Attempts > MAXRETRANSATTEMPTS

Figure 7: Flow chart of IEEE 802.11 DCF

The IEEE 802.11 DCF is implemented by the following M3_Frame_Tx class derived from the *OS_Service* base class. Only the constructor of the class is shown. The DIFS, SLOT_TIME [11] and the initial value of the contention window are assigned in the constructor as shown in Figure8.

```
M3_Frame_Tx::M3_Frame_Tx
(sc_module_name _name,
Proc_ctl_IF * host)
:OS_Service(_name){

//Initializing values of

//Contention window
CwCurrent=CWMIN;

//DIFS
DIFS=sc_time(128,SC_US);

//Slot Time
SLOT_TIME=sc_time(50,SC_US);

//Backoff time
BackOffTime=SLOT_TIME;

//Retransmission attempts
RetransmissionAttempts=0;
}
```

Figure8: The Frame Transmit Service initializing MAC parameters

In ABSOLUT the MAC protocols as well as transport-level protocols such as TCP are also modeled as servies by deriving them from the same base class *OS_Service* which provides the scheduling and synchronization mechanism. The transport-level services then request the Layer-2 services such as IEEE 802.11 for frame transmission. The do_service() method of the *OS_Service* base class is implemented by the derived class to provide the functionality of IEEE 802.11 DCF as shown in Figure 10. The do_service() method spams a separate frame transmission function for handling each request of frame transmission from the transport as shown in Figure 9.

```
void M3_Frame_Tx::do_service(){
while(true)
 {

 //Get Service attributes from OS
 m_crnt_attr=
 const_cast< Serv_attributes *>
 Current_Service_Attributes);

 //Initial retransmission Attempts
 RetransmissionAttempts=0;

 //Spawn Frame Transmission Function
 SC_FORK
 sc_spawn
 sc_bind(&M3_Frame_Tx::Tx_Crnt_Frame,
 this,static_cast<Smart_M3_Attr *>
 m_crnt_attr)))
 SC_JOIN

 //Inform OS about service completion
 m_service_complete_ev.notify();

 //Wait for next Frame Tx request
 wait();
 }
}
```

Figure 9: Implementation of Frame Transmit service. It invokes a spammed function for the transmission of a single frame.

The spammed function implements the flow chart shown in Figure 7. It attempts the retransmission transmission for up to a maximum number of retransmissions, the frame is considered lost and the transport-level service (transport protocol) is informed. For connection oriented protocols, the remaining frames are not transmitted. For connection-less protocols this information is neglected.

The transport-Level services .i.e., TCP and UDP are also modeled as *OS_Services*. Both the services make use of MAC level services for frame transmission. TCP calls the Frame Transmission service of MAC (which receives a single frame at a time for transmission) as many times as the number of frames in the transport packet. If one frame is lost (due to errors or collisions) the MAC informs transport and the rest of the frames are dropped and a packet loss is recorded. In the UDP transport-level service, the MAC does not inform the transport layer about the frame loss and all the frames are transmitted even if one or more frames of a packet are lost (due to collisions or errors). The accuracy of the modelled components is elaborated in the next section.

IV. ACCURACY OF SIMULATION RESULTS

To study the MAC protocols in isolation under a particular scenario, we abstract out the Application workloads with delays obtained after profiling or use traffic generators. Three types of traffic generators .i.e., pareto on off, exponential and constant-bit rate available in ns-2 have been integrated to ABSOLUT. The different modulation techniques like QPSK and BPSK have been modeled along-with MC-CDMA. Two channel coding techniques .i.e., convolutional and Reed Solomon codes and two channel models .i.e., binary symmetric channel and additive white Gaussian noise (AWGN) channels have been integrated by using models available in itpp library. The performance model is configured with a certain type of modulation scheme, coding scheme and channel model. Bit errors are computed using the functions available in itpp library. Frame lengths can be chosen randomly or fixed to a value before simulation to analyze MAC and transport protocols in a particular scenario.

4.1 Analysing accuracy of bit error rate calculation

Different modulation schemes available in itpp library have been used without modification. We present the results for Multi-Code CDMA with QPSK modulation. For 1e^6 bits the results are over 99.8% accurate (when compared to theoretical results) as shown in Figure 10.

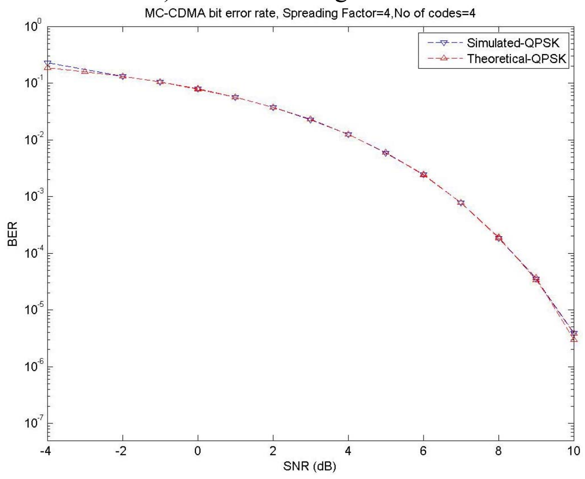


Figure 10: Theoretical versus simulation bit error rate for MC-CDMA with QPSK. Number of codes (M) = 4 .Spreading Factor (k=4). Number of bits =100,000.

4.2 Analysing accuracy of frame error rate calculation

In the absence of any encoding in IEEE 802.11, the fragment and the bit error rate are related by Equation 1.

$$P_e = 1-(1-BER)^S \quad (1)$$

Where s is the fragment size and *BER* is the Bit Error Rate and *Pe* is the probability of frame error. The bit error rates are plotted against frame error rates for different values of frame lengths as is shown in Figure 11.

The frame and bit-error rates can be recorded directly from simulation and plotted for different values of bit error rates as shown in Figure 11. The recorded simulation results are over 92% accurate when averaged after 20 simulation runs. The simulation results are compared to analytical results for Packet Lengths of 228 and 2228 as shown in Figure 11.

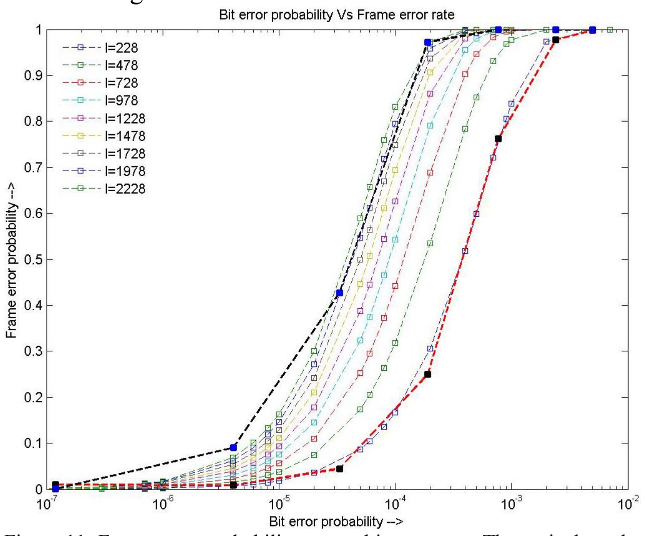


Figure 11: Frame error probability versus bit error rate. Theoretical results compared to simulation results for frame lengths 228 and 2228.

4.3 Analysing accuracy of packet error rate

In case of IEEE 802.11, one MAC service data unit (MSDU) can be partitioned into a sequence of smaller MAC protocol data unit (MPDUs) in order to increase reliability. Fragmentation is performed at each immediate transmitter. The process of recombining MPDUs into a single MSDU is called defragmentation. Defragmentation is also done at each immediate recipient. When a directed MSDU is received from the LLC with a length greater than a Fragmentation-Threshold, the MSDU is divided into MPDUs. Each fragment's length is smaller or equal to a Fragmentation-Threshold [11]. The MPDUs are sent as independent transmissions, each of which is separately acknowledged. The loss probability of transmitting a transport packet fragmented at the MAC layer into N fragments is given by the Equation 2 [12].

$$P_{wl} = 1-\left(\sum_{i=1}^{1=M} P_l^{i-1}(1-P_l)\right)^N = 1-\left(1-P_l^{M-1}\right)^N \quad (2)$$

Where P_l denotes the successful transmission probability of one attempt, i denotes the retransmission attempts and M is the maximum number of retransmission attempts. Figure 12 shows the transport packet loss rate as a function of the MAC frame loss probability during each transmission retry for a fixed number of fragments *(N=10)* and for different values of maximum retransmission attempts[12]

(M=1→10). The simulation results are compared to the analytical results as shown in Figure 12. The values of *M* and *N* were fixed, the value of signal to noise ratio (SNR) was varied and the simulation was repeated several times. The results for each value of SNR were averaged to obtain each point on the two curves. The simulation was run 20 times and the averaged results achieve an accuracy of over 85% when compared with analytical results as shown in Figure 12.

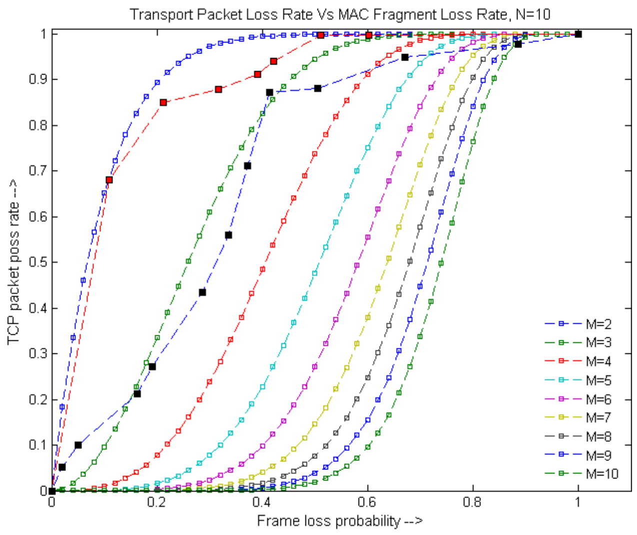


Figure 12: Theoretical versus simulation results. MAC Frame loss probability versus transport packet loss rate, for Maximum Retransmission attempts (M=2 and 3) and number of fragments (N=10).

4.4 Comparing MAC and Transport layer models with ns-2 and OMNeT++ network simulators

We now compare the throughput, Frame delays and Packet delays of ABSOLUT MAC and Transport models with ns-2 and OMNeT++ network simulators. Different Packet lengths, transport protocols and Packet transmission rates are used in both the case studies. The simulations are carried out under saturated conditions. The simulation parameters are mentioned in Table 1.

Table 1: Experiment parameters

Parameters	Values
SIFS	10 micro seconds
DIFS	50 micro seconds
Slot Interval	20 micro seconds
Preamble Length	144 bits
PLCP header Length	48 bits
Channel bit rate	2 Mbps
CWmin	32
CWmax	2048
CWo	32
EW	16

The simulations are carried out in WLAN environment in the context of M3. The abbreviation M3 means multi-device, multi-vendor, multi-domain to highlight the flexibility and portability of the technology [7]. It means that all the network client nodes called Knowledge Processors (KPs) at information level in M3 are within the transmission range of a single server called Semantic Information Broker (SIB) which acts as the only destination for the KPs.

1) Case Study 1: Comparison of simulation results at MAC and Transport Layer with ns-2

In the first case study we compare the results of our MAC and transport models with ns-2. The data traffic is generated using the Constant bit rate (CBR) traffic generator available in ns-2 simulator and the transport protocol is TCP. [12]. The traffic generators can be configured by using the simple interface as shown in Figure 13.

```
//CBR traffic generator parameters

// 2.5 milliseconds
double AverageBurstTime=.0025;
// 1 second
double InterFrameTime   = 1    ;
// 2048 bytes
double SinglePktSize=2048;
// 2 Mega bits per second
double Bitrate=2000000;

//Instantiate CBRtraffic generator
SimConfig::Instance()->SetTrafficGenerator(
CBR_TRAFFIC_GENERATOR,
AverageBurstTime,
InterFrameTime,
Bitrate,SinglePktSize
);
```

Figure 13: An example configuration of the CBR traffic generator. Packet Length=2048 Bytes. Data rate= 2 Mbps. Average Burst Time=.0025 seconds. Inter Frame Space=1 second.

With the same experimental parameters mentioned in Table 1, we perform our simulations in two different frame lengths .i.e., 512 bytes and 1024 bytes. In both the cases, the transport-level packets are not fragmented into multiple MAC frames; therefore we only use the MAC Frame delays and throughput for comparison. The average Delays for both the frame lengths are shown in Figure 14 for different number of active nodes (20→100 KPs and one SIB in smart spaces) in the network (Smart Space).

The ns-2 and ABSOLUT simulations were run 50 times and the average values were computed. The results indicate that if ns-2 is used as a reference bench mark, the results of ABOLUT Mac and transport are 70-80% accurate. The inaccuracy is due to absence of the RTS/CTS mechanism in ABSOLUT models. The results show that ABSOLUT models always produce pessimistic results, .i.e., less throughput and more delays for the same simulation scenario.

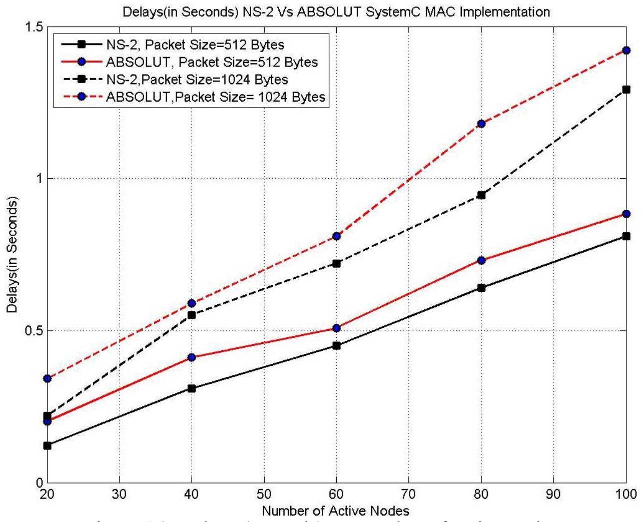


Figure 14: Delays (seconds) Vs number of active nodes (Ns-2 versus ABSOLUT)

The normalized throughput for both the frame lengths is shown in and Figure 15.

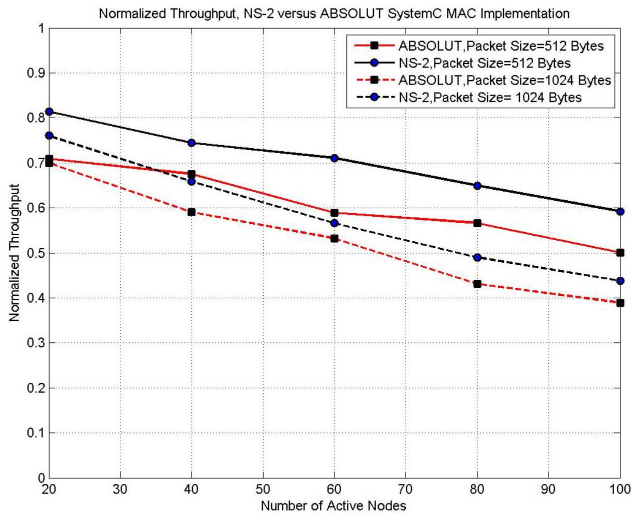


Figure 15: Normalized Throughput versus number of active nodes (Ns-2 Vs ABSOLUT)

The average collisions times (Number of collisions/100 seconds) are shown in Figure 16.

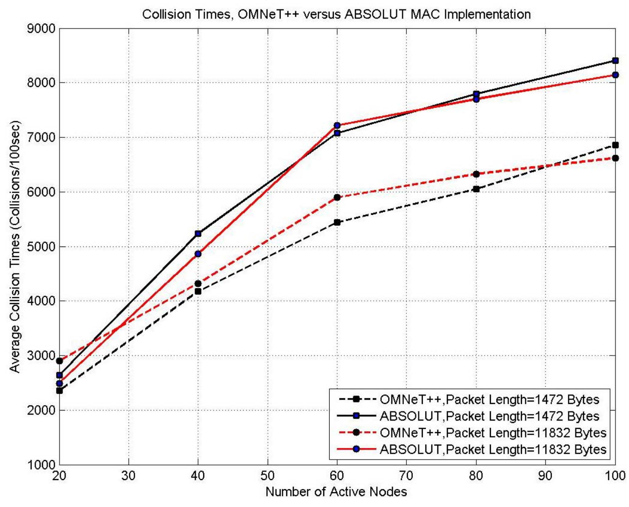


Figure 16: Average collision times Vs number of active nodes (Ns-2 Vs ABSOLUT).

2) *Case Study 2: Comparison of simulation results at MAC and Transport Layer with* OMNeT++

In the second case study, we compare the results of ABSOLUT MAC and transport models with OMNeT++. No traffic generators were used. The application sends packets at 2 milli-second interval. The simulations are performed under two scenarios. In the first scenario, the application sends 11832 Bytes long packets. The packet is fragmented into 8 fragments. As a consequence MAC sends 8 fragments, each of length 1500 Bytes. In the second scenario, the application sends a 1472 Byte packet at 2 milli-second interval. There is no fragmentation on any layer and as a result, MAC sends a single frame of length 1534 Bytes for each Application packet. Since the packet transmission rate is too fast, the collision rate is quite high which significantly increases the delays and reduces the throughput.

The maximum and average Delays for the packet length of 11832 Bytes(8 Frames/Packet) is shown in Figure 17 as the number of nodes (KPs) is varied (20→100) in the network (Smart Space).The goal is to investigate the case where multiple frames are transmitted for a single transport packet.

The OMNeT++ and ABSOLUT simulations were run 20 times and the average values were computed. The results indicate that if OMNeT++ is used as a reference bench mark, the results of ABOLUT MAC and transport are 75-90% accurate. The inaccuracy is due to the absence of the RTS/CTS mechanism in ABSOLUT models. The results show that ABSOLUT models always produce pessimistic results, .i.e., less throughput and more delays for the same simulation scenario.

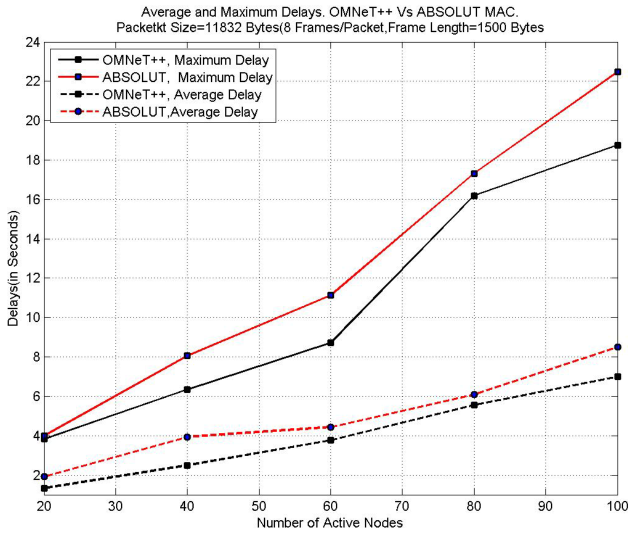


Figure 17: Maximum and Average Delays (seconds) Vs number of active nodes (OMNeT++ Vs ABSOLUT).

The normalized throughput for both the packet lengths is shown in Figure 18.

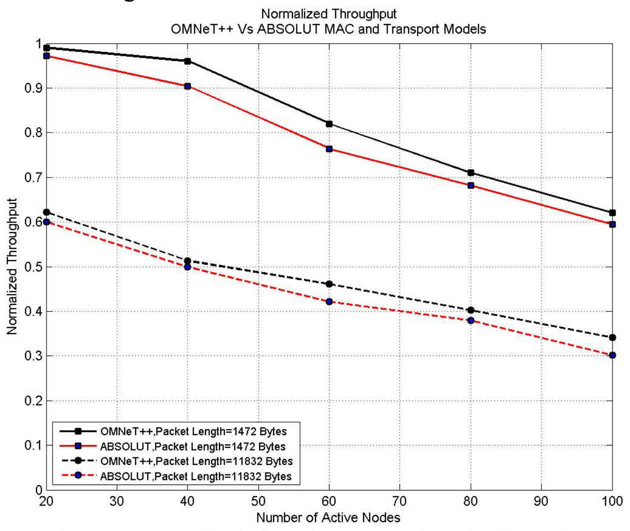


Figure 18: Normalized Throughput Vs number of active nodes (OMNeT++ Vs ABSOLUT).

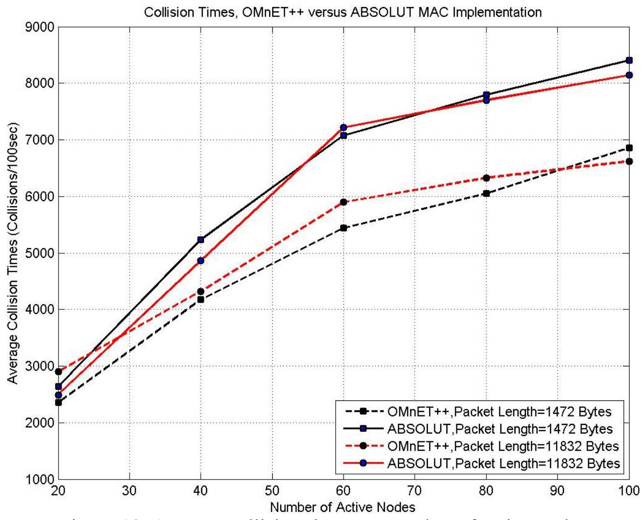
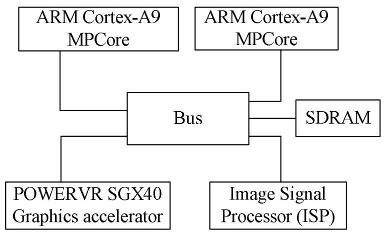


Figure 19: Average collision times Vs number of active nodes (OMNeT++ Vs ABSOLUT).

4.5 Platform utilization of Transport and MAC

Each network node (KPs or SIB) were mapped to a separate ABSOLUT platform model. Each platform model used in the both case studies is a modified OMAP-44x platform model. The MAC and Transport services were registered to the OS model of the platform. The platform model consists of two ARM Cortex-A9 processors consisting of four and three processing cores respectively instead of two (as in case of original TI OMAP44-x platform [13]), SDRAM, a POWERVR SGX40 graphics accelerator and an Image signal processor. This is shown in Figure 20. The Network-on-Chip (NoC) infrastructure was abstracted out and replaced with on-chip bus as shown in Figure 20.



ARM Cortex-A9 MPCore
ARM Cortex-A9 MPCore
Bus
SDRAM
POWERVR SGX40 Graphics accelerator
Image Signal Processor (ISP)

Figure 20: OMAP 44x Platform ABSOLUT model.

Each processor core (Cortex-A9 CPU model) has an L1 and L2 cache and can possibly share an L3 cache with one or more cores in the Multi-Core Processor model. This is shown in Figure 21.

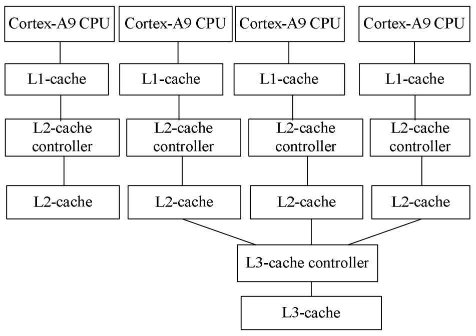


Figure 21: Diagram showing the quad-core processor model used in the performance platform model.

For performance utilization of Transport and MAC layer we only model the workload of the TCP since it is a connection-oriented, complex and more computationally intensive transport protocol then UDP. UDP is a light weight non-connection oriented transport protocol used primarily for real-time streaming applications; therefore processing costs of UDP are lower than TCP.

The workload for TCP *OS_Service* was extracted from [14]. The processing times of TCP functions were scaled down since ARM-Cortex-A9 processors operate at a much faster clock then DECstation 500/200. The approximate number of abstract instructions [2] for the TCP OS_Service processing workload were extracted and modelled as a function workload model [2]. This function workload model mimics the execution workload of the TCP transport protocol. It executes inside a Process workload model triggered by the TCP *OS_Service* during the processing of the service request. The Application models were abstracted out by using constant bit rate traffic generator and constant delays to measure the performance of TCP in isolation.

The average busy time of any processor core of any network node (KP or SIB in the case of M3) involved in the experiment was less than .00001% even for the case of 100 KPs. This is the processing time for TCP. The processing time of IEEE 802.11 is merely a subset of this processing time. The performance costs even after using NoTA DIP over TCP were found to be less than .0001% [15], thus confirming the results.

V. CONCLUSIONS AND FUTURE WORK

The analysis performed in [16] concludes that the network simulators .i.e., ns-2 and OPNET though give different absolute values under the same simulation scenarios but the trend of the results obtained from both the simulators are the same. The ABSOLUT models also validate these results. As the number of KPs is increased, OMNeT++ and ns-2 as well as ABSOLUT show a similar trend in the change in values of delays, throughput and collisions. The following conclusions can be drawn on the basis of the case studies in the previous section.

1. ABSOLUT MAC and Transport models can be used for the system-level performance simulation of distributed embedded systems. In case of distributed applications, the end-end delays and throughput play important role and must be met for providing a pleasant and acceptable end-user experience. The results are reliable since they are pessimistic when compared to state of the art network simulators for example ns-2 and OMNeT++. In other words the values of delays are always higher than OMNeT++ and ns-2. This guarantees that if the ABSOLUT MAC and transport models validate the values of delays and throughput, they are definitely validated by OMNeT++ and ns-2. Thus the modelled MAC and Transport layer services can be confidently used for system-level performance simulation of distributed applications and architectural exploration.
2. The application models can be replaced by traffic generators or constant delays to study the behaviour of Transport and MAC protocols in isolation. This helps to identify the potential bottlenecks at different layers in the OSI model giving full coverage of the OSI protocol stack in

architectural exploration of distributed embedded systems.

3. The performance costs of the MAC and transport layers are negligible and can be abstracted out for system-level performance simulation.

4. In the case of M3 architecture, according to ABSOLUT MAC and transport models, the following conclusions can be drawn in the traffic conditions considered in the two case studies. In first case study, the average delay per packet increases from .24 and .33 seconds to .9 and 1.39 seconds and the normalized throughput decrease from .71 and .7 to .5 and .41 for packet lengths 512 and 1024 bytes. In second case study, the average delay per packet increases from 2 and 4 seconds to 8.1 and 23 seconds and the normalized throughput decrease from .98 and .6 to .6 and .4 for packet lengths 111832 and 1472 bytes respectively. The platform utilization is negligible in both cases. Therefore IEEE 802.11 MAC and TCP do not show any significant performance bottlenecks as far as platform utilization is concerned but the delays increase significantly as more and more KPs join the smart space under the considered traffic conditions. The throughput also decrease significantly as the number of KPs increases in a smart space under these traffic conditions. Hence we conclude from our analysis that for small scale smart spaces such as smart cars and class rooms M3 will operate pretty well with IEEE 802.11 standard operating at Layer-2 and TCP. On the other hand in large scale smart spaces such as smart hockey stadiums operating 1000s of KPs to share information, IEEE 802.11 has to be either optimized or replaced by another potential solution at MAC-layer.

In the future, it is planned to further extend the ABSOLUT methodology to the incorporate multithreading application workload modelling support and C++ workload extraction methodology. These extensions will enable the seamless integration of design and performance simulation of distributed applications. The extended ABSOLUT framework will then employed for the system-level performance simulation of distributed GENESYS, NoTA and M3 applications. In case of non-distributed GENESYS applications that milestone has already been achieved [3].

VI. ACKNOWLEDGEMENTS

This work was performed in the Artemis SOFIA project partially funded by Tekes – The Finnish Funding Agency for Technology and Innovation and the European Union. The work was performed in cooperation with Finnish ICT SHOK research project Future Internet. Authors would like to thank all their colleagues for valuable discussions about the topic.

REFERENCES

[1] K. Keutzer, A. Newton, J. Rabaey and A. Sangiovanni-Vincentelli,"System-level design: orthogonalization of concerns and platformbased design", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 19 (12), 2000, pp. 1523 - 1543.

[2] J. Kreku, M. Hoppari, T. Kestilä, Y. Qu, J.-P. Soininen, P. Andersson and K. Tiensyrjä. Combining UML2 Application and SystemC Platform Modelling for Performance Evaluation of Real-Time Embedded Systems. Hindawi Publishing Corporation. EURASIP Journal on Embedded Systems. Volume 2008, Article ID 712329, 18 pages, doi:10.1155/2008/712329.

[3] Linking GENESYS Application Architecture Modelling with Platform Performance Simulation. Subayal Khan, Susanna Pantsar-Syväniemi, Jari Kreku, Kari Tiensyrjä and Juha-Pekka Soininen. 12th Forum on specification and Design Languages, FDL2009. Sep 22- 24, 2009. Sophia Antipolis, France.

[4] S. Khan et al., "From y-chart to seamless integration of applicationdesign and performance simulation," in System on Chip (SoC), 2010International Symposium on, 2010, pp. 18 –25.

[5] http://www.systemc.org/home/

[6] http://itpp.sourceforge.net

[7] Lappeteläinen, Antti et. al., 'Networked Systems, Services, and Information - The Ultimate Digital Convergence'. 1st International Conference on Network on Terminal Architecture, June 11, 2008, Helsinki.

[8] http://www.isi.edu/nsnam/ns/

[9] http://www.omnetpp.org/

[10] B. Kienhuis, E. Deprettere, K. Vissers and P. van der Wolf. Approach for quantitative analysis of application-specific dataflow architectures," in Proceedings of the IEEE International Conference on Application- Specific Systems, Architectures and Processors (ASAP '97), pp. 338– 349, Zurich, Switzerland. July 1997. USC Center for Software Engineering, Guidelines for Model-Based (System) Architecting and Software Engineering, http://sunset.usc.edu/research/MBASE, 2003.

[11] J.M. Paul, D.E. Thomas, and A.S. Cassidy. High-Level Modeling and Simulation of Single-Chip Programmable Heterogeneous Multiprocessors. ACM Transactions on Design Automation of Electronic Systems, Vol. 10, No. 3, 2005, pp. 431-461.

[12] Fethi Filali, Impact of Link-Layer Fragmentation and Retransmissions on TCP performance in 802.11-based Networks. in Proc. of MWCN 2005, 7th IFIP/IEEE International Conference on Mobile and Wireless Communications Networks, September 19th-21st 2005,Marrakech, Marrocco.

[13] www.ti.com

[14] Jonathan Kay, Joseph Pasquale: The Importance of Non-Data Touching Processing Overheads in TCP/IP. SIGCOMM 1993: 259-268

[15] Instantiation and feasibility evaluation of NoTA SOAD via MARTE profile and binary instrumentation. Khan, Subayal; Tiensyrjä, Kari; Nurmi,Jari. The 7th International Conference and Expo on Emerging Technologies for a Smarter World. CEWIT 2010. Incheon, 27 - 29 Sep. 2010

[16] ns-2 vs. OPNET: a comparative study of the IEEE 802.11e technology on MANET environments, P. Pablo Garrido, Manuel P.Malumbres and Carlos T.Calafate. SIMUTOOLS 2008 - 1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems.