A SIMPLE AND EFFICIENT SOLUTION TO HALF-FRAME PIPELINING FOR MODULAR SOFTWARE DEFINED RADIO

Arnd-Ragnar Rhiemeier, Timo Weiss, Friedrich K. Jondral
Universität Karlsruhe (TH), Institut für Nachrichtentechnik, Karlsruhe, Germany
{rhiemeier, weiss, jondral}@int.uni-karlsruhe.de

ABSTRACT

The understanding of fundamental approaches to designing and operating Modular Software Defined Radio (Mod-SDR) are of immediate relevance for offering flexible services to mobile users, by means of a single communication device. We review Half-Frame Pipelining (HFP) as a way of operating any software defined PHY layer on a specific multiprocessor hardware architecture. In this paper HFP is improved against former results with respect to the runtime efficiency of its partitioning concept and the achievable speedup. A comparison to another approach, Graph Duplication Pipelining (GDP), reveals HFP's advantages and disadvantages. PHY layer signal processing for both circuit-switched and packet-switched services is discussed in detail. Finally, we conclude on the utility of HFP for application in Mod-SDR terminals.

Keywords – Modular SDR, allocation of computing resources, PHY layer firmware, multiprocess synchronization

1. MODULAR SOFTWARE DEFINED RADIO

The notion of software in a Software Defined Radio environment often remains in the vague and may even be employed to summarize all sorts of executable code across all layers of a terminal. Most notably, however, physical (PHY) layer signal processing imposes the most severe constraints on the execution of code: Hard real-time physical signal processing as opposed to a growing share of best effort computing on higher layers of the protocol stack.

We focus on a centralized, non-preemptive operating system (being a vital part of the terminal's firmware) that administers the execution of software modules for any software defined PHY layer of a wireless terminal. The concept of Mod-SDR has been introduced [1][2] in order to derive general design guidelines under these circumstances. Throughout its lifetime the operating system will encounter a variety of software modules and processing runtimes. We claim that this variety is so vast that it can be modeled as a random process. Furthermore, the generalized modeling of SDR software includes random graphs [3]. The latter serve as an abstract, structural representation of wireless communication standards, where nodes are software modules and intermediate computation results flow along directed edges.

In order to understand the fundamental design principles of Mod-SDR systems, we have restricted the number of processors to $L = 2$. In this paper we continue to study Mod-SDR using our stochastic linear resource-runtime model [4] with Gaussian probability density function (pdf), rectangularly windowed over $[c_{min}; c_{max}] = [0.5; 1.5]$, and pdf parameters $\mu_c = 1.0$, $\sigma_c = 0.25$. The case $L = 2$ may seem to represent too low a number of processors, but that number will not be excessive in mobile terminals anyway, and mastering the case $L=2$ is necessary before looking at any more general case.

2. CIRCUIT-SWITCHED SERVICES

We have introduced HFP [3] as a method for scheduling PHY layer software modules, with a focus on circuit-switched services. The basic idea of HFP is to divide a graph into a left and a right partition, in order to allow one processor to execute code of the second half of some radio frame n, while the other processor already executes code of the first half of its successor frame $(n+1)$.

2.1. Partitioning for Half-Frame Pipelining

In [3] we have proposed to solve the HFP-associated partitioning problem by a spectral approach, based solely on a composite edge cost involving nodes' B^* values as well as the original potential link cost. Unfortunately, we observed that this procedure occasionally results in non-vertical cuts, so post-processing is necessary. Furthermore, we have introduced a load balancing procedure which was based on nodes' B values [3]. The best speedup results on random graphs were obtained by this combination of spectral partitioning and B level load balancing, but it remained questionable whether there was a more straightforward solution to HFP partitioning for Mod-SDR. Especially the need for *two* post-processing steps following a method as elegant and fast as spectral partitioning appeared to be inappropriate and too time-consuming.

Proceeding of the SDR 04 Technical Conference and Product Exposition. Copyright © 2004 SDR Forum. All Rights Reserved

In a first attempt to improve the overall runtime for partitioning we have determined B^* levels for all nodes in both forward and reverse direction. Then, prior to spectral partitioning, we have averaged the forward and the (inverted) reverse B^* levels in order to capture the graph structure before and after a specific node, but to no avail: The two post-processing steps did remain necessary to guarantee a vertical cut.

As a consequence we became aware of the fact that the spectral method merely provides a starting partition and that the observed speedup results can mainly be ascribed to the load balancing procedure. Then we may as well run the iterative load balancing procedure starting from a trivial partition (where all nodes except for the target node initially belong to one partition), thus saving the runtime of the entire spectral approach. Furthermore, B level load balancing is based on selecting the node with maximum B level for transfer to the smaller partition. This certainly guarantees a vertical cut, but maximum B level is only a sufficient, not a necessary condition: *All* nodes with outgoing edges pointing to nodes of no other than the smaller partition are eligible for a transfer. Those include but are not limited to the maximum B level node.

Another drawback of B level load balancing is its inability to balance transferred node runtime against additional link cost incurred by that transfer. In contrast, the Kernighan/Lin (KL) set exchange procedure, which we studied earlier in the context of Mod-SDR [4], takes into account both runtime and link cost. Therefore, it is an ideal candidate to replace B level load balancing.

2.2. Algorithm Timing Results

Figure 1 shows the different algorithm runtimes (CPU time) as a function of relative bus speed β. The constant β determines how much faster a processor can transfer a certain data block over a bus rather than produce the same data block as the output of a module [3]. The measurements are represented in the form $\hat{\mu} \pm \hat{\sigma}$, where $\hat{\mu}$ is the estimated average algorithm runtime, $\hat{\sigma}$ is an estimate for the standard deviation

$$\hat{\sigma} = \sqrt{\frac{1}{K-1} \cdot \sum_{k=1}^{K} (p_k - \hat{\mu})^2}$$

and the sample size is $K = 2000$ per β. We recognize that trivial partitioning + KL load balancing is about five times faster than the initial approach if bus transfers dominate the modular system (slow buses, $\beta \ll 1$), and it is about twice as fast for fast buses ($\beta > 1$). Note that the standard deviation is *not* due to measurement errors in the simulation code acquiring the CPU time, but caused by algorithm response to the random graphs. This becomes all the more evident as we look at the increased $\hat{\sigma}$ values under KL partitioning around $\beta = 1$, where bus transfer and signal processing node runtimes fall in the same order of magnitude.

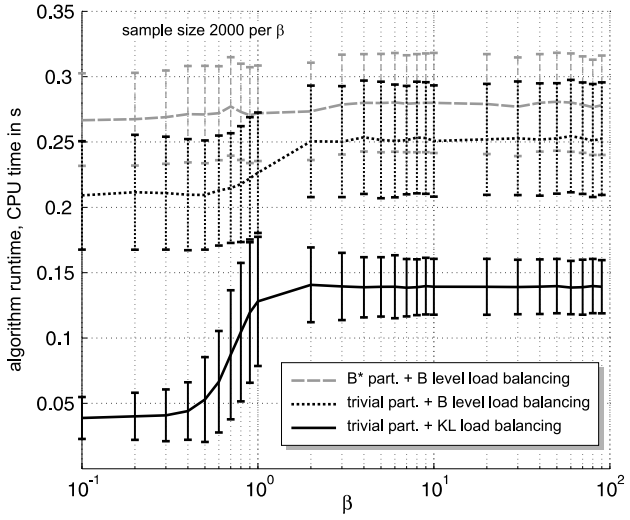

Fig. 1. Partitioning algorithm runtimes

2.3. Speedup Results

Figure 2 shows the achievable fraction of maximum speedup [5] as a function of β for both the former approach (dashed contour lines only) and trivial partitioning + KL load balancing (dots for speedup measurements, and solid contour lines for the 5%, the 50%, and the 95% quantile. Note that β values supporting the contour lines are more closely spaced than the log grid where curvature is high.) We observe that the *simpler and faster* KL method (with the same effect, namely partitioning for HFP) outperforms our initial partitioning and load balancing approach: As indicated by the solid median contour line the majority of Mod-SDR realizations remains above $0.5\,\overline{s}$, even for low values of β. More interesting for the practical application of HFP, however, is the high β region, where speedup approaches its limit $\overline{s} = 1$ closer and in a more condensed way. Moreover, if we move towards lower values of β (for power saving reasons, for example), the speedup degradation is more graceful than before.

2.4. Comparison to Graph Duplication Pipelining

We have seen that HFP is suboptimal regarding speedup and that speedup loss against the upper bound $\overline{s}$ depends on β. In contrast, an optimal scheme such as GDP [6] reaches $\overline{s}$ for all β. The basic idea of GDP is the following: Duplicate the entire graph and assign a complete copy to each processor. This way, partitioning is not an issue at all, and producing a dense static schedule is quite easy [6].

A major drawback of GDP, however, is its increased memory demand due to the duplication process: For each copy of the graph both input (I) and output (O) memory must be allocated in the I/O space where the PHY layer hardware system interfaces to other parts (analog RF frontend, service access points of higher layers) of the signal processing system.

Proceeding of the SDR 04 Technical Conference and Product Exposition. Copyright © 2004 SDR Forum. All Rights Reserved

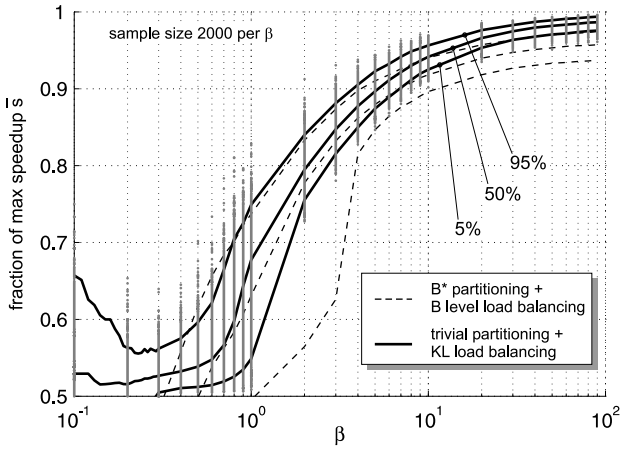

Fig. 2. Half-Frame Pipelining, speedup as a function of relative bus speed

HFP, on the contrary, allocates the I/O space only once because it operates on a single copy of the graph. Nevertheless, it requires some system-internal shared memory for the transfer of intermediate results between the processors, in the form of a separate hardware component. This means an increased hardware design effort, but it *does not* worsen HFP's overall memory budget. GDP also needs such memory space, just in the form of distributed memory (associated to every processor) and – again – twice as much of it.

To sum up, we can state that HFP's main advantage is moderate memory demand, which comes at the price of some speedup loss against the theoretical upper bound $\overline{s}$. To keep this loss small, HFP requires a fast, power-inefficient system bus. Therefore, *only if* memory demand proves to be a critical design issue, HFP may be competitive against GDP for circuit-switched services.

3. PACKET-SWITCHED SERVICES

So far, we have only discussed PHY layer signal processing for circuit-switched services. However, there is a growing interest in packet-oriented services, especially as radio communication devices and laptop computers merge into wireless personal digital assistants and as the Internet Protocol (IP) continues to represent a major portion of all data traffic.

3.1. WLAN Example

As an introductory example, let us consider the transmitter of an OFDM-based wireless LAN standard such as IEEE802.11a. Its PHY layer signal processing system transforms incoming data bits (including both user data and higher layer control data) into OFDM symbol packets that are transmitted over the antenna. The creation of a single OFDM symbol involves computations which can be captured in a directed graph, in the same way as before. To form a complete packet of N OFDM symbols (or of N "radio frames", to be more general) all graph computations need to be repeated N times.

3.2. Generalization

It is important to notice that both the order of execution of software modules and all their processing runtimes remain *unchanged* during these repetitions. The processing just transforms different bits into adjacent (but otherwise independent) radio frames. Furthermore, be aware of a basic property of communication standards: For any kind of communication to be successful, all communication partners must respect common signal processing steps, or, in other words, agree on a common *standard*. Possibly, quality of service (in the form of bit error rates, or other measures) may still be negotiated between the partners, but once communication starts, both intra-packet and intra-standard signal processing is deterministic. We do not see any reason for some statistical demand for processing power [7] on the PHY layer of a Mod-SDR. The only intra-standard parameter that *will* change over time is the packet size N. Whenever small packet sizes are dominant, the filling and the emptying of the software pipeline will cause runtime overhead and hence reduce speedup. It remains to be discussed how to operate best a Mod-SDR terminal offering packet-switched services to the user.

3.3. Simulation of Packet-Oriented Wireless Standards

In the following, the transmitter of a packet-oriented wireless communication standard is considered. An equivalent line of arguments can be constructed for the receiver side of such a standard.

First of all, we assume that higher layer data bits of an entire packet are available at the input of the PHY layer signal processing system, so that input data processing can be triggered at arbitrary, equidistant, or recurrent non-equidistant instants in time. This assumption is not at all far-fetched since, in comparison to the output memory, the input memory required to store input bits at the system interface is relatively small. This is because both data representation word length and the number of samples per radio frame tend to grow as signal processing advances across the PHY layer towards the analog RF frontend.

Second, in order to render the operating system firmware as simple as possible, we assume that *both* processors are exclusively reserved for PHY layer signal processing as soon as bit 1 of frame 1 is input to the system. Likewise, both processors are released only after the last output sample of frame N has left the system.

3.4. Results for Half-Frame Pipelining

Figure 3 shows the scheduling scheme for HFP and N radio frames per packet. Processor activity is plotted over time. The proposed order of execution of regular signal processing nodes (P-nodes), bus transfer nodes (B-nodes), and input/output nodes (I-/O-nodes) is given in detail in the enlarged elliptical area, which is a zoom into the steady state of the HFP schedule. The filling and the emptying of the

Proceeding of the SDR 04 Technical Conference and Product Exposition. Copyright © 2004 SDR Forum. All Rights Reserved

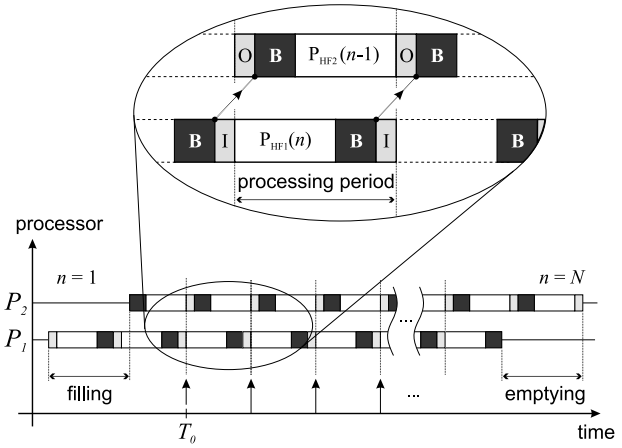

Fig. 3. HFP, scheduling for packet processing, N frames per packet

pipeline can be readily identified in the figure. In the sequel, the timing reference for output processing be the execution of the first O-node per radio frame. Trigger instants are indicated by an impulse train along the time axis, and the first trigger instant is marked by T_0. All following timing-related figures are organized this way. Figure 3 shows that output processing triggers for packet-oriented HFP are equidistant, as a matter of principle.

Originally, the absolute speedup s is defined as the reservation time of a mono-processor divided by that of a multi-processor. In the sequel, however, we prefer to use the relative speedup $y = s/\overline{s}$ [5]. Figure 4 shows the simulation results for HFP and some small numbers of frames per packet, $2 \le N \le 7$. For reasons of legibility, only the contour lines of the 5%, the 50%, and the 90% quantile are plotted. Depending on the packet size N, it is easy to derive an upper bound on the relative speedup from Fig. 3 by letting $\beta \to \infty$ (all bus-related execution times approach zero):

$$y_{\text{HFP}} = s_{\text{HFP}}/\overline{s} < \frac{N}{N+1}$$

Figure 4 also shows these upper bounds as dashed horizontal lines.

3.5. Results for Graph Duplication Pipelining

Figure 5 shows a scheduling scheme that represents the fastest possible way of completing N frame computations per packet under the GDP framework. We observe that the filling and the emptying of the pipeline consume very little time, but the trigger instants are non-equidistant.

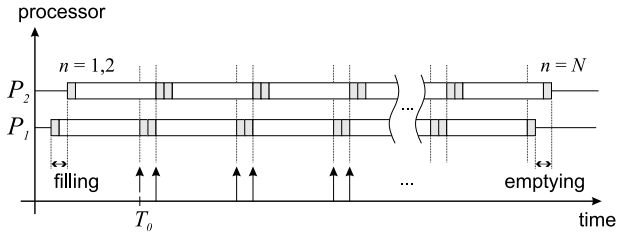

Fig. 5. GDP, non-equidistant trigger instants, high speedup

Figure 6 shows the speedup results for GDP, the above scheduling scheme and $2 \le N \le 7$. It can be shown [6] that here the relative speedup is bound *from below* by $N/(N+1)$, and that this bound is indeed achieved for odd numbers of frames per packet (see $N \in \{3; 5; 7\}$ in Fig. 6) and reasonable bus speed β.

Continuous transmission of every packet by the analog RF frontend requires the PHY layer to support equidistant output processing triggers. If we enforce such triggering, we necessarily increase GDP's memory demand: Figure 7 shows that new output memory for some frame $(n+1)$ needs to be allocated *before* the RF transmission of frame $(n-1)$ has even been started. Therefore, the output memory spaces of these two frames co-exist, in addition to the anyway separate

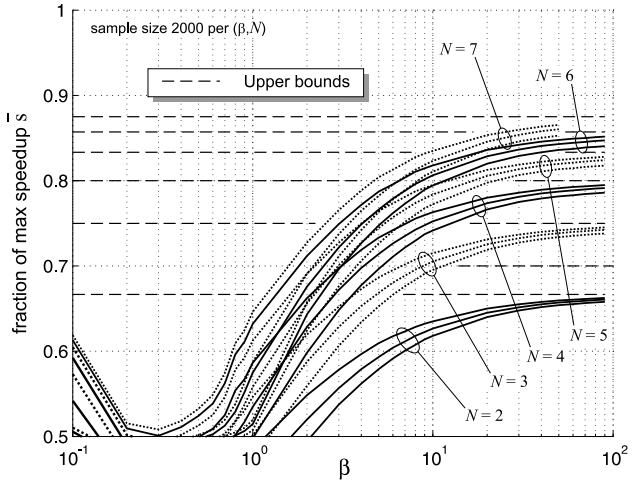

Fig. 4. HFP, packet processing, speedup as a function of relative bus speed

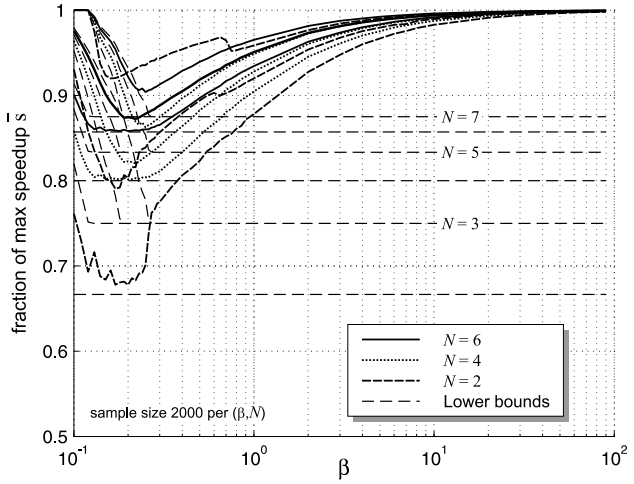

Fig. 6. GDP, packet processing, speedup as a function of relative bus speed

Proceeding of the SDR 04 Technical Conference and Product Exposition. Copyright © 2004 SDR Forum. All Rights Reserved

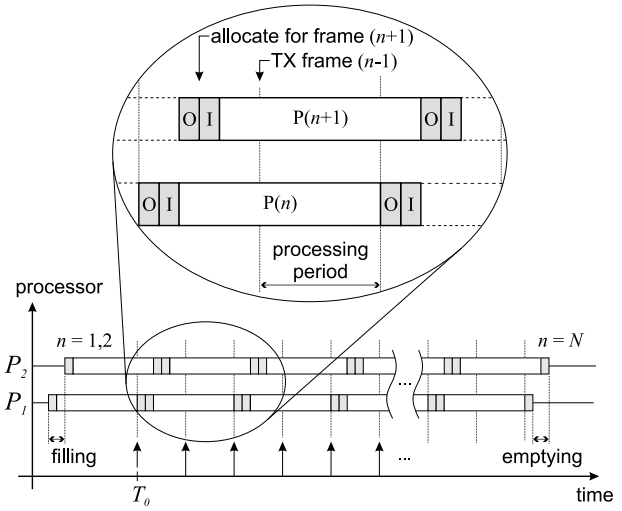

Fig. 7. GDP, enforced equidistant trigger instants, high memory demand

output memory space for frame n computed on the other processor. We have already pointed out in section 3 that output memory is likely to be greater than input memory at the transmitter. Therefore, we certainly expect GDP with enforced equidistant trigger instants to create significant memory overhead.

An alternative solution to both sustaining GDP *and* supporting equidistant triggers is sketched in Fig. 8. All computations executed on the second processor are time-shifted to become aligned with equidistant trigger instants. This way, the output memory can be re-allocated for frame $(n+1)$ *after* the RF transmission of the previous frame $(n-1)$ has been completed.

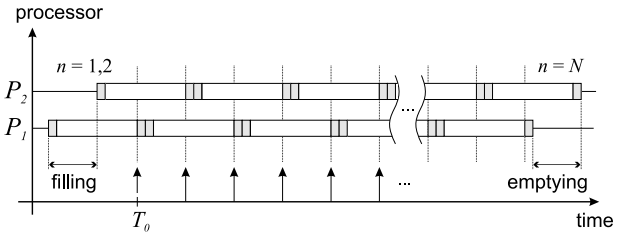

Fig. 8. GDP, equidistant trigger instants, reduced speedup

A drawback of this alternative is a reduction in speedup. It is easy to derive from Fig. 8 that now the achievable fraction of maximum speedup is exactly $N/(N+1)$ for all $N \geq 2$, independent of β. That is to say: We achieve no more speedup than indicated by the dashed lines in Fig. 6. Nonetheless, those limits are *upper bounds* for packet-oriented HFP. Therefore, we conclude that all proposed GDP scheduling schemes outperform HFP under the described conditions of packet processing.

4. RELEVANCE TO ESTABLISHED WLAN STANDARDS

The expected average speedup can only be predicted if packet size statistics of some tangible WLAN implementation are known. Hitherto, we have only shown speedup contributions for small N. A naive working assumption could be that packet size among established WLAN standards is much larger than $N = 7$ anyway, because short packets naturally imply relatively large overhead (and, of course, small speedup). If packet size was large anyway, our discussion of speedup results for small N would be irrelevant. In order to reveal that short packets are indeed of practical relevance to established WLAN implementations, a packet transmission scenario based on the IEEE802.11a standard [8] has been simulated (see Fig. 9).

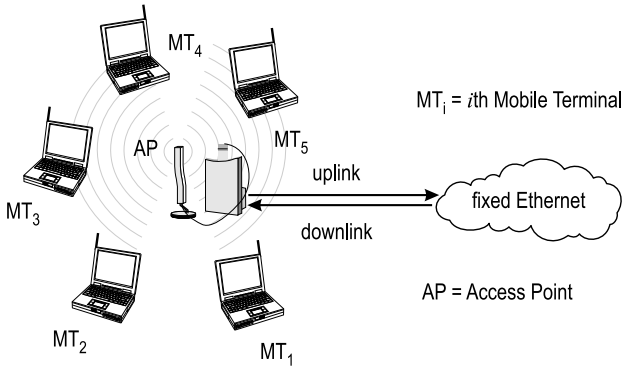

Fig. 9. WLAN simulation scenario

The required IP traffic model has been adopted from [9]. It is an integral model that does not distinguish between service types. An uplink/downlink traffic ratio of 1/4 has been assumed, and the number of mobile terminals (MTs) around the access point (AP) has been set to five. Both the MAC layer and the PHY layer have been implemented at a high degree of detail. All functions like error control, beacon transmission, hidden station avoidance, channel coding etc. are considered. The PHY layer operates in 64-QAM at a code rate of $R = 3/4$ whenever the MAC layer delivers *data packets* for processing. MAC layer *control packets* (such as RTS, CTS, ACK, etc.) are transmitted using BPSK modulation, although QPSK and 16-QAM are admissible, too [8]. Packet size histograms have been generated for an indoor channel [10] and a vehicle-to-vehicle channel [11] with a coherence time of $T_C \approx f_{D,max}^{-1} = 6$ ms for both of them.

Figure 10 shows the simulation results for the indoor channel and average signal-to-noise ratios ranging from 10 dB to 30 dB. The packet size is given in multiples of OFDM symbols, *excluding* the preamble of length 16 μs (the length of 4 OFDM symbols) and the SIGNAL field (1 OFDM symbol). Certainly, the traffic model includes IP packets of up to 1500 bytes (larger packets close to 40 symbols and 60

Proceeding of the SDR 04 Technical Conference and Product Exposition. Copyright © 2004 SDR Forum. All Rights Reserved

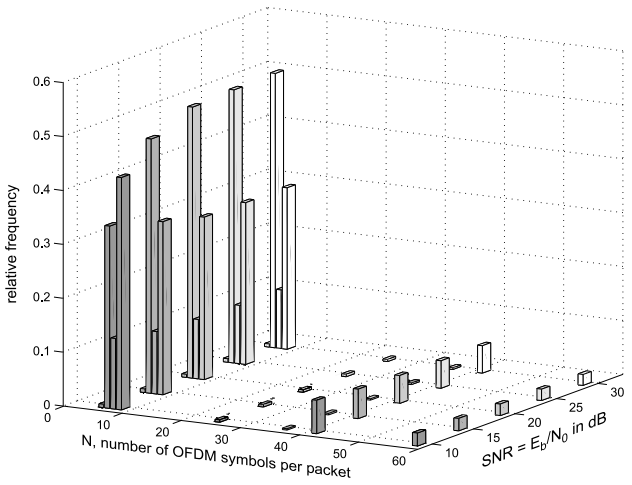

Fig. 10. Packet size histograms, IP traffic, indoor channel

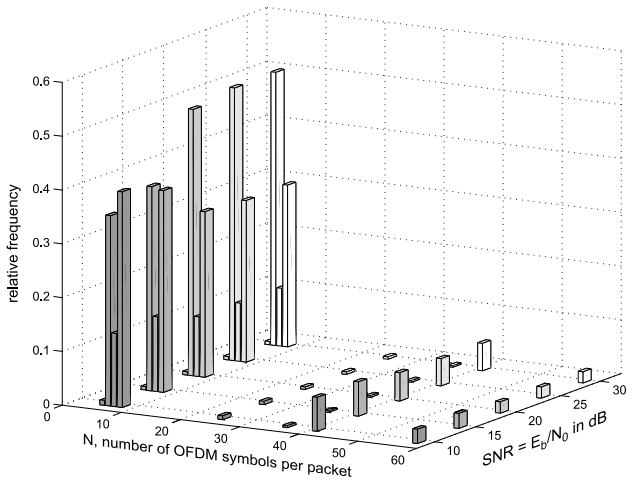

Fig. 11. Packet size histograms, IP traffic, vehicle-to-vehicle channel

symbols), but regardless of the average SNR, we observe a massive occurrence of transmissions with less than 10 OFDM symbols. With increasing SNR the relative frequency of occurrence of packet size $N = 5$ grows, and that of size $N = 7$ shrinks, while $N = 6$ as well as all larger packets remain virtually unaffected by the average SNR of the channel. Results for the vehicle-to-vehicle channel (see Fig. 11) exhibit a very similar packet size histogram pattern.

A theoretical projection may be that a high level of SNR represents a "best case", if small N are a design concern, because low SNRs would certainly imply more packet errors and thus lead to an increase in the number of short MAC layer control packets. This is true indeed, but we can give a more precise interpretation here based on our simulations: $N = 5$ corresponds to Clear To Send (CTS) and Acknowledgement (ACK) packets of size 14 bytes each, while Request To Send (RTS) packets of size 20 bytes are transformed into $N = 7$ OFDM symbols. Of course, the PHY layer traffic is a mixture of MAC layer data *and* control packets, and their individual contributions are indistinguishable in the integral model. Nevertheless, we will argue later why the data packet contributions are more or less independent of SNR. For the time being, let us think of these contributions as a constant (per N) histogram floor, which is deductible from the integral histogram measurement. As a consequence, the remainder must represent MAC layer control packets.

Now, if we are at a high level of average SNR, the control packet size ratio of $(N = 5) : (N = 7)$ is about $2 : 1$. This fits our expectation that "for good channels" nearly all data transmissions (of arbitrary size) are successful, preceeded by an RTS/CTS handshake and followed by an ACK. If we go towards lower SNR, however, all packets become increasingly vulnerable to packet errors. Especially, if RTS fails, the handshake is physically interrupted by the channel, and the transmitter just re-attempts an RTS after some timeout delay. This way the portion of RTS increases relative to all other packet types, but notably against CTS and ACK (see both Fig. 10 and Fig. 11).

Finally, data packets are virtually unaffected by the average channel SNR, because their transmission is *conditional* on the success of the RTS/CTS handshake. Even if the average channel SNR is low, the *instantaneous* SNR may be high. Should the initial RTS/CTS handshake prove successful, then such a high instantaneous SNR is very likely to be sustained over the entire data packet transmission time. Even the longest IP packets (1500 bytes) transform into PHY layer packets of size $N = 56$ OFDM symbols (224 μs), plus the preamble and the SIGNAL field, then totaling 244 μs. A complete RTS/CTS/data/ACK sequence [8] amounts to less than 480 μs, which is much shorter than the channel coherence time of $T_C = 6$ ms. In other words, data packet transmissions only take place when the channel is successfully probed to be good by the RTS/CTS handshake.

5. CONCLUSION

We have briefly reviewed the concept of Mod-SDR and the modeling of PHY layer software by directed graphs. In a first step, HFP has been devised as a method for scheduling software modules if the PHY layer is to support circuit-switched services. We have refined our approach to graph partitioning for HFP in terms of both algorithm runtime and speedup. Then we have compared HFP to GDP with respect to speedup and memory demand. Based on these considerations and on the computer simulations presented in section 2 we can draw the following conclusions:

- HFP is suboptimal regarding speedup, and it requires high bus speeds β to keep speedup loss against the optimum small.
- HFP for circuit-switched services is systematically outperformed by GDP that, in turn, requires more memory than HFP.

Proceeding of the SDR 04 Technical Conference and Product Exposition. Copyright © 2004 SDR Forum. All Rights Reserved

- The most simple and effective solution to HFP is Kernighan/Lin partitioning, starting from a trivial partition configuration.

In a second step, we have analyzed packet-oriented HFP, the underlying motivation being a growing interest in packet-switched services and networks, which Mod-SDR terminals must support seamlessly. We have presented the scheduling scheme of HFP along with speedup results for packet computations and an upper bound, depending on the number of radio frames per packet. The subsequent analysis of GDP scheduling has turned up three options with mutually exclusive advantages: High speedup, equidistant trigger instants, or moderate memory demand. The simulation results have shown that GDP speedup is *at least* $N/(N+1)$. Based on the results of section 3 we can state the following:

- HFP inherently supports equidistant output processing triggers and requires less memory than GDP which can only trade either speedup or non-equidistant triggering for memory.
- Still, packet-oriented HFP is systematically outperformed by GDP, regardless of both GDP scheduling options and packet size statistics.

Finally, we have shown that small numbers of radio frames per packet (and thus the way of operating Mod-SDR) are indeed of practical relevance to established WLAN standards. For the example of our IEEE802.11a scenario packets with a small number of OFDM symbols do easily amount to a great share among all PHY layer packets, not depending too much on the channel type and average SNR, as we have discovered in section 4.

We conclude that, in general, HFP can merely prove competitive against GDP if memory is a critical issue for the design of a Mod-SDR terminal. Otherwise, GDP is to be preferred when offering either circuit-switched or packet-switched services to the mobile user.

6. REFERENCES

[1] A.-R. Rhiemeier and F. K. Jondral, "Mathematical modeling of the software radio design problem," *IEICE Trans. Commun.*, vol. E86-B, no. 12, pp. 3456–3467, Dec 2003, special issue on SDR Technology.

[2] ——, "On the design of modular software defined radio systems," in *IEE Colloquium on DSP Enabled Radio*. Alba Campus, Livingston, Scotland (UK), Sept 2003.

[3] A.-R. Rhiemeier, "A comparison of scheduling approaches in modular SDR," in *3rd Karlsruhe Workshop on Software Defined Radios (SDR'04)*. Karlsruhe (Germany), March 2004, ISSN 1616 - 6019.

[4] A.-R. Rhiemeier and F. K. Jondral, "A software partitioning algorithm for modular software defined radio," in *6th Int'l Symp. Wireless Personal Multimedia Communications (WPMC'03)*. Yokosuka (Japan), Oct 2003.

[5] U. Berthold, A.-R. Rhiemeier, and F. K. Jondral, "Spectral partitioning for modular software defined radio," in *IEEE Semiannual Vehicular Technology Conf. (VTC'04) Spring*. Milano (Italy), May 2004.

[6] ——, "A pipelining approach to operating modular software defined radio," in *IEEE Sarnoff Symposium (SARNOFF'04)*. Princeton, NJ (USA), April 2004.

[7] J. Mitola, "Software radio architecture: A mathematical perspective," *IEEE J. Select. Areas Commun.*, vol. 17, no. 4, pp. 514–538, 1999.

[8] "ANSI/IEEE Std 802.11, Wireless LAN MAC and PHY specifications," 1999 Edition, and IEEE 802.11a-1999, High-speed Physical Layer in the 5GHz Band.

[9] J. Weinmiller, M. Schlager, A. Festag, and A. Wolisz, "Performance Study of Access Control in Wireless LANs - IEEE 802.11 DFWMAC and ETSI RES 10 hiperlan," *Journal on Mobile Networks and Applications*, vol. 2, no. 1, June 1997.

[10] "Universal Mobile Telecommunications System (UMTS); UMTS Terrestrial Radio Access (UTRA); Concept Evaluation (UMTS 30.06; TR 101 146)," Tech. Rep., Dec 1997.

[11] J. Maurer, T. M. Schaefer, and W. Wiesbeck, "Wave Propagation Modelling for Inter-Vehicle Communications," in *URSI 27th General Assembly, Paper-No. 1231, Maastricht (The Netherlands)*, Aug 2002.

Proceeding of the SDR 04 Technical Conference and Product Exposition. Copyright © 2004 SDR Forum. All Rights Reserved