TSG-RAN Working Group 1 R1-01-0048

Boston, U.S.A. January 15-18, 2001

Source: Motorola

Title: Clarifications on Dual-Channel Stop-and-Wait HARQ

1.0 Introduction

This document clarifies the purpose, possible configurations, and signaling requirements of Dual-Channel Stop-and-Wait Hybrid ARQ (DC-SW-HARQ).

2.0 Purpose

Hybrid ARQ can significantly increase user throughput over independent ARQ and FEC. In effect, Hybrid ARQ adapts to the channel by sending additional increments of redundancy, which decreases the overall coding rate and effectively lowers the data rate to match the channel.[1] In addition to improving throughput, Hybrid ARQ can be viewed as an enabler for adaptive modulation and coding (AMC). Hybrid ARQ makes AMC rate selection more robust because it does not rely on channel estimates to provide the correct code rate, but instead relies on the errors signaled by the ARQ protocol. However, Hybrid ARQ requires additional receiver memory to store unsuccessful attempts. As a result, Hybrid ARQ can significantly increase UE complexity depending on the ARQ protocol design and the network configuration.

Window based Selective Repeat (SR) is a common type of ARQ protocol employed by many systems including RLC R99. SR is generally insensitive to delay and has the favorable property of repeating only those blocks that have been received in error. To accomplish this feat, the SR ARQ transmitter must employ a sequence number to identify each block it sends. SR may fully utilize the available channel capacity by ensuring that the maximum block sequence number (MBSN) exceeds the number of blocks transmitted in one round trip feedback delay. The greater the feedback delay the larger the maximum sequence number must be. However, when Hybrid ARQ is partnered with SR, several difficulties are seen.

- *UE memory requirements are high*. The mobile must store soft samples for each transmission of a block. MSBN blocks may be in transit at any time. A large MBSN requires significant storage in the UE adding to the unit's cost.
- *Hybrid ARQ requires the receiver to reliably determine the sequence number of each transmission*. Unlike conventional ARQ, every block is used even if there is an error in the data. In addition, the sequence information must be very reliable to overcome whatever channel conditions have induced errors in the data. Typically a separate, strong code must be used to encode the sequence information, effectively multiplying the bandwidth required for signaling.

Stop-and-wait is one of the simplest forms of ARQ requiring very little overhead [1]. In stop-and-wait, the transmitter operates on the current block until the block has been received successfully. Protocol correctness is ensured with a simple one-bit sequence number that identifies the current or the next block. As a result, the control overhead is minimal. Acknowledgement overhead is also minimal, as the indication of a successful/unsuccessful decoding (using ACK, NACK, etc) may be signaled concisely with a single bit. Furthermore, because only a single block is in transit at a time, memory requirements at the UE are also minimized. Therefore, HARQ using a stop-and-wait mechanism offers significant improvements by reducing the overall bandwidth required for signaling and the UE memory. However, one major drawback exists: acknowledgements are not instantaneous and therefore after every transmission, the transmitter must wait to receive the acknowledgement prior to transmitting the next block. This is a well-known problem with stop-and-wait ARQ. In the interim, the channel remains idle and system capacity goes wasted. In a slotted system, the feedback delay will waste at least half the system capacity while the transmitter is waiting for acknowledgments. As a result, at least every other timeslot must go idle even on an error free channel.

---

[1] 'Hybrid ARQ' (or HARQ) is defined as any combined ARQ and FEC method that saves failed decoding attempts for future joint decoding. Hybrid ARQ therefore encompasses many named variants, such as code combining, Chase combining, incremental redundancy (IR), partial IR, full IR, type II HARQ, type III HARQ, etc, Dual-Channel Stop-and-Wait Hybrid ARQ applies to all forms of Hybrid ARQ.

Therefore, an ARQ method for Hybrid ARQ with the minimal complexity of stop-and-wait but with the throughput efficiency of SR is desired.

3.0 Dual-Channel Solution

Dual channel stop-and-wait Hybrid ARQ offers a solution by parallelizing the stop-and-wait protocol and in effect running a separate instantiation of the Hybrid ARQ protocol when the first channel is idle. As a result no system capacity goes wasted since one instance of the algorithm communicates a data block on the forward link at the same time that the other communicates an acknowledgment on the reverse link[2].

The dual channel stop-and-wait ARQ places requirements both on the UE and the Network. Section 3.1 will examine the UE processing requirements by looking at transmissions to a single device. Section 3.2 takes the devices described in Section 3.1 and places them in a multi-user configuration to understand how the dual channel stop-and-wait concept works in a system.

3.1 Single Device Example

Figure 1 considers the case where a single user is using the channel. In Figure 1, the system consists of a single source and destination device over a slotted data channel. For this example, one slot is equivalent to one HSDPA TTI where it is assumed that all HSDPA TTIs are of the same size. The data channel is divided into even and odd timeslots to identify the independent instances of the Hybrid ARQ protocol. The even or odd state of the channel may be signaled explicitly on the control channel or derived globally from system information such as the CFN. The least significant bit of the CFN would be enough to identify the even or odd channel. Data blocks arrive from the network and are queued at the source. The source employs a dual channel sequencer to admit data blocks to either the even or odd transmitter. Once admitted, each transmitter performs a conventional stop-and-wait ARQ algorithm in its respective timeslot by transmitting the data block on the data channel and sequence bit on the associated control channel. Similar to the source, the destination device contains both an odd and even receiver, receiving blocks from the respective odd and even timeslots. Note, that these receivers are only a logical representation of a single hardware receiver. Each receiver is coupled with an independent Hybrid ARQ decoder. The Hybrid ARQ decoder signals the success (or failure) of the data block on a separate feedback channel. Both even and odd feedback channels exist to support each independent instance of the stop-and-wait.

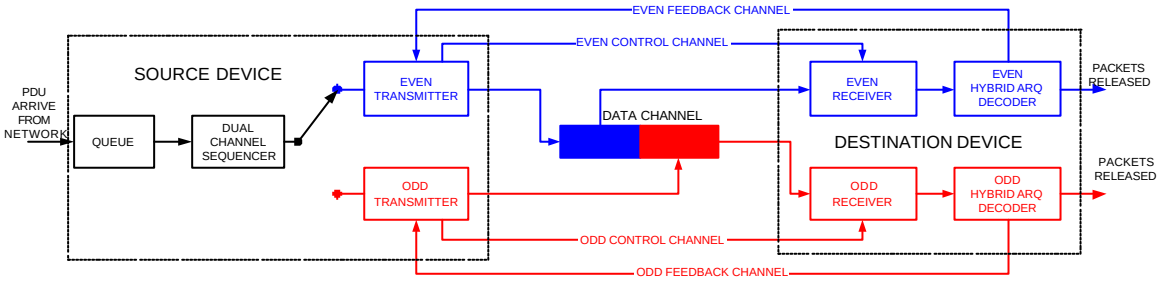

Figure 1 Single device block diagram.

In summary, this configuration requires:

1) A downlink slotted data channel.

2) A mechanism to associate each slot with either the even or the odd instance.

---

[2] At first glance, dual-channel stop-and-wait may appear identical to SR with a window size of two. However, they are very different. Advancing a transmission window in SR requires that all blocks in the window have been transmitted successfully. A persistent failure in the transmission of one of the oldest blocks, early in the sequence, will prevent the ARQ window from advancing and prevent all communication on the channel. In dual-channel stop-and-wait, a persistent failure in one transmission block will only affect communication on one of the channels, allowing data transmission to continue on the other. Since the SR system must increase the window size in order to maintain an equivalent stall probability to dual-channel stop-and-wait, the SR system will require much more UE memory to achieve similar performance.

3) An even/odd downlink control channel to identify the sequence number of the even/odd stop-and-wait data PDU.

4) An odd downlink control channel to identify the sequence number of the odd stop-and-wait data PDU.

5) An even uplink feedback channel to signal either success or failure of the even downlink data channel.

6) An odd uplink feedback channel to signal either success or failure of the odd downlink data channel.

Figure 2 illustrates the timing of a dual channel stop-and-wait scheme showing how the even and odd data channels, control channels and feedback channels can be interleaved in time. In one slot, the even control and data are sent on the downlink while the odd feedback channel is sent on the uplink. In the next slot, the roles are reversed where the odd control and data are sent on the downlink while the even feedback channel is sent on the uplink.

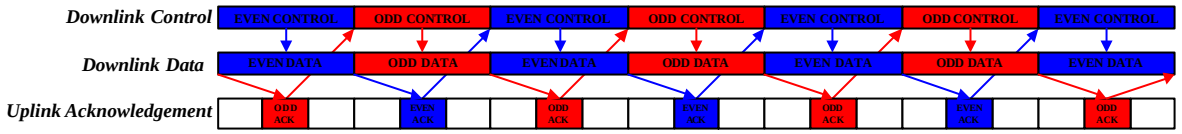

Figure 2 Timing of dual-channel stop-and-wait

In this single device example, the destination is required to maintain no more than two buffers of data[3], one for each instance of the stop-and-wait protocol. Timing depicted in Figure 2 suggests that the acknowledgment on the uplink channel does not occupy a complete slot allowing for processing of the downlink data by the destination device and processing of acknowledgements by the source device. In practice, it is possible to increase the processing times for both the source and destination devices by increasing the number of channels from 2 channels to N channels. As a result, the memory requirements at the destination devices would increase in proportion to the number of channels. Figure 3 and Figure 4 show the device and timing block diagrams, respectively, when the number of channels has been increased to four.

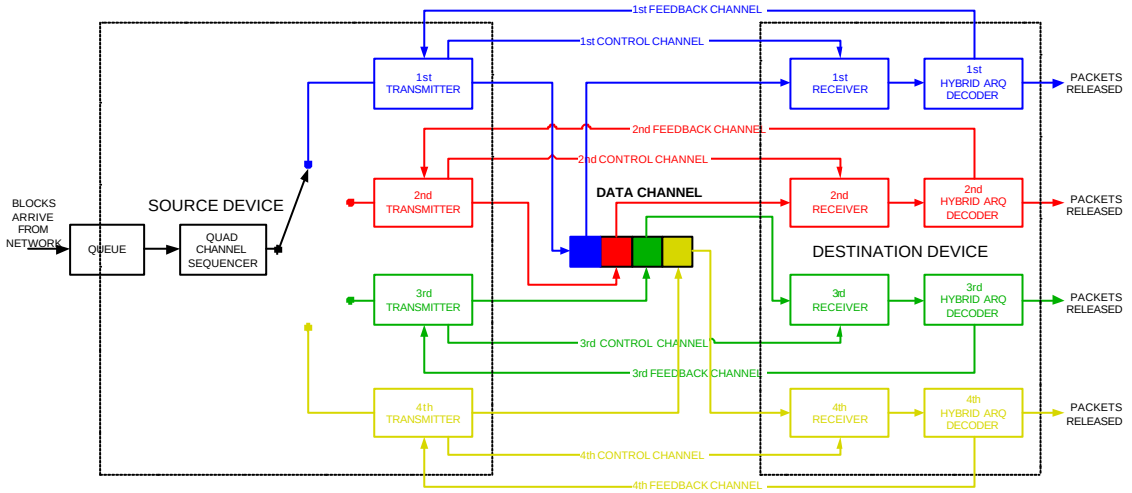

Figure 3 Single device example block diagram with quad channel stop-and-wait.

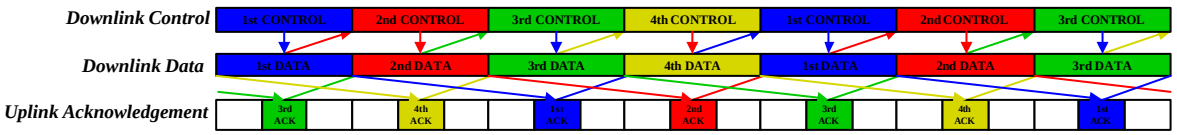

Figure 4 Timing of quad-channel stop-and-wait

[3] At most two *sets* of buffers for some types of Hybrid ARQ, such as incremental redundancy.

3.2 Multi-device Example

Figure 5 extends the system described in section 3.1 to multiple sources and destinations. The source and destination devices are identical to Figure 1 and are coupled by independent downlink control and uplink feedback channels. Unlike Figure 1, all source and destination pairs share a single downlink data channel. Therefore, a system scheduler is required to arbitrate between the multiple sources in the system. The scheduler will select which source owns the current timeslot based on the status of each individual queue or possibly one large combined queue. In this case, each source must signal both the channel state, even or odd, and the ownership identity of the timeslot to the destination over the control channel. In particular, the presence of multiple devices allows the scheduler to defer retransmissions until a device is experiencing acceptable channel conditions, while maintaining full system utilization. In all other aspects, the system performs as in the single user case.

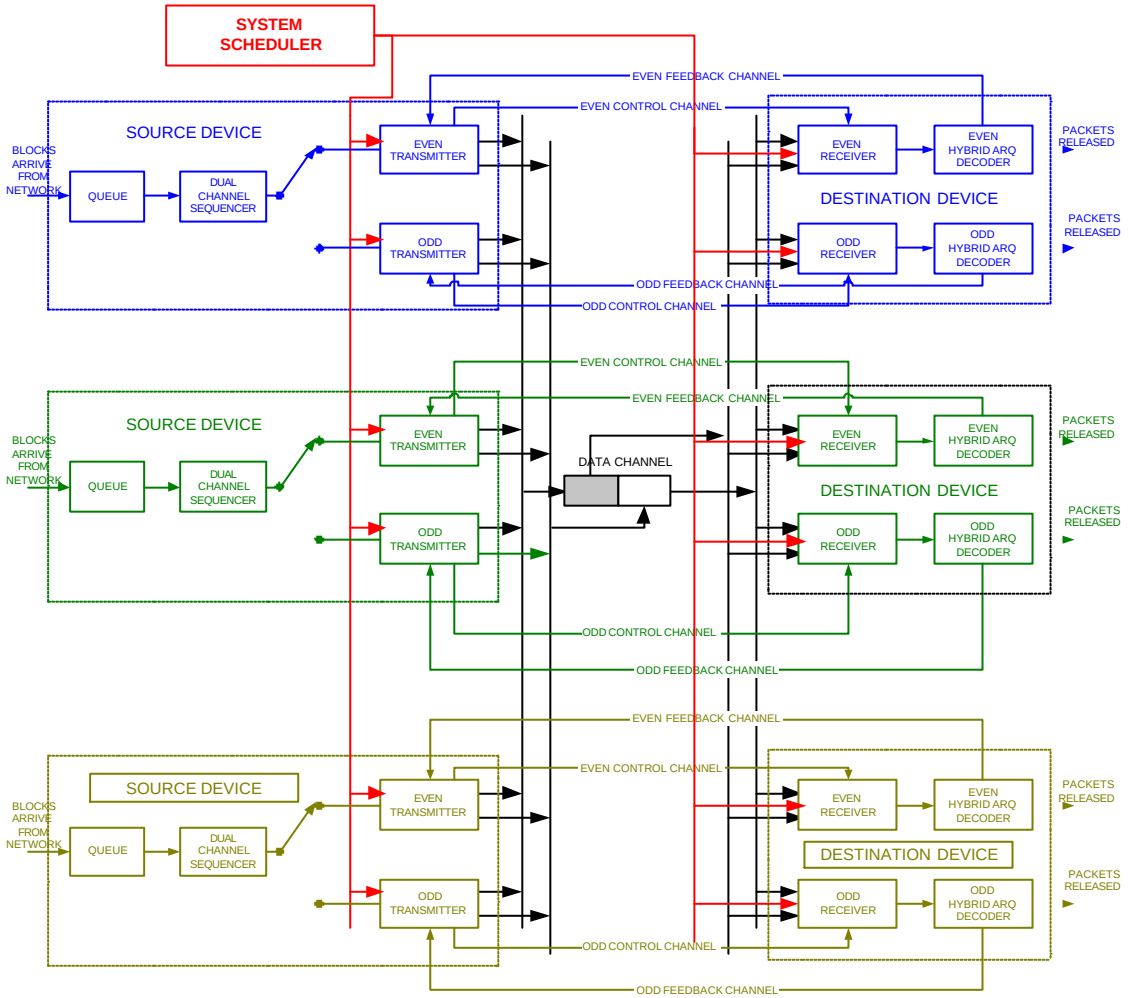

Figure 5 Multi-device/single data channel block diagram.

The multi-device example has nearly the identical requirements to the single device example. As before the following are required per device:

1) A downlink slotted data channel.
2) A mechanism to associate each slot with either the even or the odd instance.
3) An even downlink control channel to identify the sequence number of the even stop-and-wait data PDU.
4) An odd downlink control channel to identify the sequence number of the odd stop-and-wait data PDU.
5) An even uplink feedback channel to signal either success or failure of the even downlink data channel.
6) An odd uplink feedback channel to signal either success or failure of the odd downlink data channel.

In addition, the multi device example requires:

7) An addressing control channel from the system scheduler to each of the destination devices.

4.0 Signaling Requirements

This section discusses the signaling required to support the dual-channel stop-and-wait protocol based on the concepts described in Section 3. All downlink control signaling could be supported by a low-rate dedicated control channel similar in concept to the DCH associated with DSCH. However, due to the low-latency requirements it is advantageous to terminate the control and data channels at the Node B instead of the RNC [2]. All users would share a single downlink HS-DSCH that may be code multiplexed among multiple users by assigning an integer number of multi-codes spread at a fixed spreading factor. Finally, a low-rate feedback channel is assumed to carry the acknowledgement information.

4.1 Addressing

A HS-DPCCH can be configured as a simple Boolean indicator channel identifying whether an assignment exists in the current frame. When a channel is assigned, the HS-DPCCH carries supplemental information identifying the particulars of the channel assigned. The assignment identifies:

1) The number of multi-codes assigned. Requires 5 bits if each multicode has SF=32.

2) The starting address of the base code. Requires 5 bits if each multicode has SF=32, assuming that an assignment of multiple codes is logically contiguous.

The supplemental assignment information could be carried on a single HS-DPCCH.

4.2 Channel Sequencing

The stop-and-wait channels may be identified either implicitly or explicitly.

Implicit identification would tie the even and odd channels instances to the cell-specific frame timing or CFN. However, if the TTI is less then 15 slots (i.e. one frame) an offset counter into the frame will be necessary to identify the channel. The structure of the offset counter would depend on the TTI and number of channels. For example a TTI equal to 5 slots combined with 2 channel stop-and-wait protocol would require that the offset counter be reset every 30 slots or 20 ms to ensure an equal occurrence of each channel. Alternatively, a TTI equal to 5 slots combined with a 3 channel stop-and-wait protocol would allow the offset counter to be reset every 15 slots or 10 ms. In either case, the identification of the channel state would not require additional signaling. The number of channels and TTI could be determined a priori and maintained independently by the network and the UE.

Explicit signalling would require that an additional field be added to the supplemental signaling channel to identify the channel state. This channel state field would only need to be sent when a data PDU was sent. The channel state field would be $\log_2 N$ bits where N represents the number of N-channel stop-and-wait ARQ.

Of the two options, the implicit signaling appears the most favorable since it reduces the number of signaling bits required on the supplemental signaling channel. While explicit signaling does offer greater flexibility in the scheduling of retransmissions, it is unlikely that this additional flexibility will outweigh the greater signaling burden.

4.3 Acknowledgements

The Hybrid ARQ receiver must positively acknowledge or negatively acknowledge each data PDU sent on the signaling channel. There are many ways to configure the acknowledgement channel. Three methods are:

1) Send a single bit for each attempted TTI where a one represents a positive acknowledgement and a zero represents a negative acknowledgement. The signaling would be similar to the TPC bits on the physical layer.

2) Only send a negative acknowledgement if the attempt was not successful. No message implies a positive acknowledgement.

3) Only send a positive acknowledgement if the attempt was successful. No message implies a negative acknowledgement.

5.0 Conclusions

This document discussed the purpose, possible configurations, and signaling requirements of Dual-Channel Stop-and-Wait Hybrid ARQ (DC-SW-HARQ). Specifically, it identified that Hybrid ARQ channels are associated with the UE receiver. It showed that the dual-channel concept can be extended to the *N*-channel stop-and-wait ARQ. Finally, it discussed some of the signaling necessary to support *N*-Channel Stop-and-Wait Hybrid ARQ.

References

[1] D. Bertsekas, R. Gallager, Data Networks Second Edition, Prentice Hall, Englewood Cliffs, New Jersey
[2] Ericsson, Motorola, R2-A010010, "HSDPA Radio Interface Protocol Architecture"