ME218C - 2020 Communications Protocol

Welcome to the Internet of Pings

Revision History:

Rev #	Date	Comments
0	5/11/2020	First draft of the protocol based on initial discussions
1	5/11/2020	Restructured the TDMA format after discussions with the teaching staff
2	5/12/2020	Another overhaul after class review
3	5/13/2020	Addition of timing diagram, examples, and additional clarity
4	5/15/2020	Addition of dB levels for input. Clean up of explanations in preparation for final release.
5	5/17/2020	Corrected error in checksum calculation 1 in the examples
6	5/19/2020	Added clarification on how NEXT_TURNs and SYNCs are counted for transition into the corresponding next game states
7	5/22/2020	Addition of requirement to disable notification sounds with push to talk

Hardware Requirements:

Each station in the game shall use the provided speaker. The frequencies chosen were based on testing performed on said speaker. This will help maintain continuity between teams when it comes to transmitting. When driving the speaker, it is recommended that the signal has a peak to peak voltage below 1.5V to prevent any damage to the speaker during operation. Each station shall use the provided electret microphone for reception of audio signal. Again, testing was done using this microphone. Each station must also use a servo to autonomously press a key on the computer using the push to talk functionality of the video conferencing platform.

Speaker output should be tested with this website. The decibel level should be between 110 dB and 120 dB. Choice of microphone input to the computer is open, the only requirement is the dB levels the computer reads is within the aforementioned range.

Physical Layer:

Encoding:

Each team shall implement frequency modulation to send dibit symbols over the audio line. Table 1 specifies the frequencies to be used and their corresponding dibit encoding.

Table 1. Dibit frequency encoding

Period (us)			Frequency (Hz)			Dibit
MIN	***TYP***	***MAX***	***MIN***	***TYP***	***MAX***	**-**
990	1000	1010	990.10	1000.00	1010.10	00
790	800	810	1234.57	1250.00	1265.82	01
590	600	610	1639.34	1666.67	1694.92	10
390	400	410	2439.02	2500.00	2564.10	11

Note that the encoding is done by period rather than frequency. The tolerances for the frequency are calculated based on the tolerances for the periods. Units are expected to follow the period tolerance strictly. If this is done, the frequency tolerance will be held as well. The above periods were chosen since they reside in the flat area of the frequency response of the speaker. Furthermore, these periods result in frequencies that are within the range of human speech, thus they will hopefully be transmitted better over the internet as most communications platforms are designed for human speech.

The transfers shall occur with the MSB being transmitted first, much like the I2C protocol. Therefore, when transferring a byte with this physical protocol, bits <7:6> shall transmit first, and bits<1:0> shall transfer last.

Audio communication will take place over the audio app Discord and the settings will be as described in the project specifications. In addition, the notification sound of activating push to talk must be disabled. This can be done through Settings > Notifications > Sounds, and unchecking “PPT Activate” and “PPT Deactivate”. The audio application used may change if Discord proves to be unreliable.

Timing Information:

Each team shall communicate at 10 baud (dibits/s), resulting in a total bit rate of 20 bits/s.

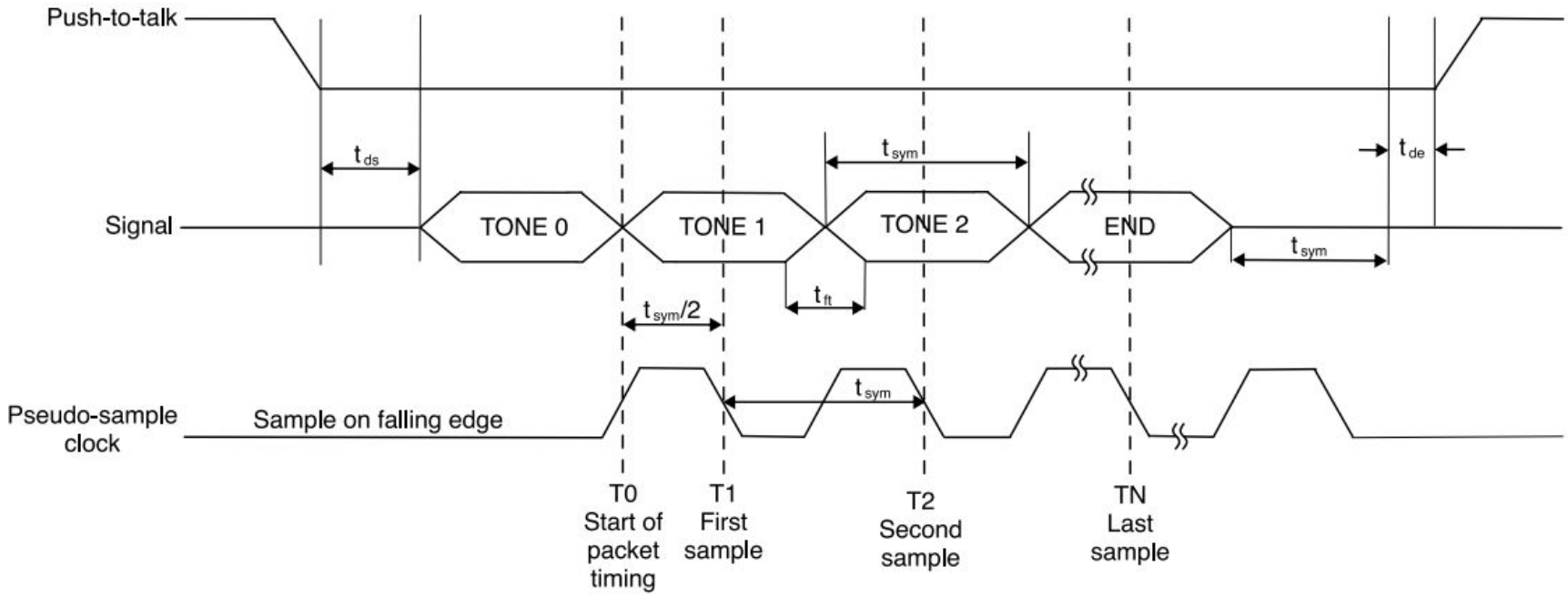


Figure 1. Timing Diagram of packet transmission

Figure 1 shows a pseudo-sample clock. This is for illustrative purposes only as the communication system is asynchronous with an agreement that baud rate shall be 10 tones/s. The sample clock shown is a recommended internal clocking that stations should use to sample the tones as they come in from the line. Sampling in this implementation to acquire the correct tone must occur at least 5 ms from the nominal symbol transition and is recommended to occur at the center of the tone.

Note that the T0 event does not occur on the first tone, but the transition between TONE 0 and TONE 1. This is considered the start of the packet and where all sampling shall begin. This is a strict requirement as it will be used for syncing purposes. More details of this can be found in the Syncing section of the Data Layer -> Media Access section.

The audio software being used to transmit signals must incorporate a push to talk feature. Push to talk means that audio cannot be transmitted unless a key of the users choice is depressed on their keyboard. This key must be depressed by a servo motor controlled by the station and Figure 1 demonstrates when the push to talk should be enabled. The push to talk line shows when the audio line is actually open to the transmitter, NOT when the servo starts to move down. Since various keyboards have various throws, each station shall determine the time it takes from servo starting to move to the audio line being open for them to transmit. This is to ensure that no two stations are generating audio simultaneously. The t_{ds} and t_{de} specifications are for when the line is already open and ready for transmitting. When using the Push-to-talk feature in Discord, the time delay should be set to 0ms.

Below is a table for the various specifications on the timing.

Table 2. Timing parameters.

Parameter	Description	MIN	TYP	MAX	Units
t_{sym}	Time for each symbol	99.5	100	100.5	ms
t_{ds}	Delay time from when the push-to-talk key is pressed and the unit begins sending tones.	10	-	100	ms
t_{de}	Delay time after transmission is complete and the push-to-talk key is lifted.	10	-	100	ms
t_{ft}	Time between tone transitions	-	-	50	us

*Tolerances are defined based on the limitations of the framework timer.

"Some ships are designed to sink… others require our assistance."
--Nathan Zelk, former RM2(SS) USS Montpelier

Data Layer:

Framing:

Below is a frame diagram of how the data shall be sent..

Table 3. Framing diagram demonstrating the purposes of the data dibits; highlighted frames are included in the ChkSum calculation

Activity	*Start Delim*	*Turn*	Address			*Activity/ Heading*	*Data*		*ChkSum*	*Quiet Time*	*TOTALS*		
			MSB	*Source Sub*	*Source Stat.*		*MSN*	*LSN*			*Data*	*Dibit Time (ms)*	*Tx Time (s)*
Dibit Index	0	1	2-3		4	5-6	7-8	9-10	11	12	13		
#Bits	2	2	1	3	2	4	4	4	2	2	26		
Values	3	0-2	1 = OWN 0 = ENEMY	n = Team n	0 = CONN 1 = SONAR 2 = TORPEDO	See Activities Table	See Activities Table	See Activities Table	= 0x3 - sum modulo 4	--		100	1.3

Turn - The Turn Dibit specifies which turn the game state is currently in for added redundancy

ChkSum - To calculate the checksum, all of the dibits marked in blue are summed as a 2 bit value; any overflow over 2 bits causes the value to loop back to 0. This value is subtracted from 0x3 to yield the correct checksum value. The receiver should calculate the checksum and compare it with the received checksum in dibits <13:14> to validate that there were no errors in the receiving process. See Example 1 in the appendix for an example of calculating the checksum.

Activity/Heading - Refers to the type of transmission being sent. See table 4 for the valid activities in the game.

Data - This is used for any data that is required along with an activity (see Table 4 for more details). For inter-submarine transfers, the protocol specified shall be followed so every submarine knows how to interpret the data. For internal submarine communication (between stations of the same sub), any protocol can be used, as long as it fits within 4 dibits (1 byte).

Activity Set Summary:

This table holds the list of all possible activities and what data needs to be put in the data section to communicate the activity to others.

General Format for Activities:

Table 4. Activities set at a glance

Action/Event	Activity/Heading				Data MSN				Data LSN				Addr MSB
	11	10	9	8	7	6	5	4	3	2	1	0	19
PING_OMNI	1	1	1	1	Y <7:4>				X <3:0>				0
POS_UPDATE[1]	1	HEADING [3]			Y <7:4>				X <3:0>				1
PING_DIRECTED[2]	1	HEADING [3]			Y <7:4>				X <3:0>				0
ECHO	1	0	0	0	Y <7:4>				X <3:0>				0
HIT	0	1	1	1	Y <7:4>				X <3:0>				1,0
POS_REQUEST	0	1	1	0	0x00				0x00				1
DETONATE	0	1	0	1	Y <7:4>				X <3:0>				0
LAUNCHED	0	1	0	0	Y <7:4>				X <3:0>				0
SYNC	0	0	1	1	0	0	0	0	0	0	0	0	0
NO_ACTION	0	0	1	0	0	0	0	0	0	0	0	0	0
NEXT_TURN	0	0	0	1	TURN NUMBER								0
RIP	0	0	0	0	1	1	1	0	0	1	0	0	0
[1] If source sub is own sub's address AND Source MSB == 1													
[2] If source sub is an enemy sub's address AND Source MSB == 0													
[3] Up: 001, RightUp: 010, RightDown: 011, Down: 100, LeftDown: 101, LeftUp: 110													

Activity Descriptions:

PING_OMNI - Sends a short range ping in all directions. Submarines in range are to respond.

POS_UPDATE - This is an internal communication between stations and therefore the addr MSB needs to be set. The heading is included in the Activity section. The heads are standardized as follows:

001 Up	110 Left Up
010 Right Up	101 Left Down
011 RightDown	100 Down

The data attached to the position update are the X and Y coordinates. The values of X and Y to follow are laid out on figure 1 of the project specification. Since the Z coordinate can be calculated, it does not need to be included

PING_DIRECTED - This is an external communication between submarines. The heading is incorporated into the activity section. The headings are defined the same as in POS_UPDATE. Note, the MSB == 0. If the MSB == 1 this becomes a POS_UPDATE activity.

ECHO - When a submarine is PINGed, it shall send back an ECHO. This is the traditional use of the ECHO activity. When hearing an ECHO, the submarine needs to remember what kind of PING it sent and only display valid PINGs back to the user. This is an honor system. See “Self-Enforced Rules” in the Appendix for more details. In this manner, only 1 ECHO needs to be sent even if a SONAR received multiple PINGs.

HIT - This signifies the submarine has been hit. When Addr MSB == 1 this is an internal communication but when MSB == 0 this is a broadcast. Stations are allowed to listen to a HIT even if the MSB is 0 since it will indicate to them their sub is compromised.

POS_REQUEST - When a station drops out, it can send this activity to get back the position update from the team’s other stations

DETONATE - This signifies the explosion of a torpedo. Teams are required to accurately respond to this. There is no enforcement in the protocol to check if a submarine is in a location where a torpedo detonates.

LAUNCHED - Torpedo has been launched. All submarines in the arena can hear it and determine the location.

SYNC - This activity is used at the start of game play to allow the stations to enter the game, see the syncing portion of the Media Access section for more details on syncing. Syncing shall only be used at the start of the game, but after all stations in a single TDMA frame have synced, the game starts and the activity should not be used again.

NO_ACTION - A station shall send this activity if its user inputs an IDLE for their turn or if they do no action within the time allotted. A NO_ACTION shall also be used during human time. See the Network Layer for more details.

NEXT TURN - This action conveys to the other stations that the station hasn't disconnected from the network and that it is ready for the next turn in the game. Only when all stations display this action will the game state update to the next turn. The exception is that if some stations become disconnected due to rough wifi seas, the game will loop through the stations one more time for the station to come back online and then the game will then increment itself to the next turn in the game. See more details in the Data Flow section of the Network Layer.

There may be cases in which a sub would transmit NEXT_TURN but would need to respond to an event from a subsequent sub. In this case, the sub would transmit the responses necessary and then transmit NEXT_TURN once all relevant events have been resolved. Only when all stations (or all connected stations if this is the additional loop for checking NEXT_TURNs) in a single TDMA frame are transmitting NEXT_TURN, does the turn increment.

"The cure for anything is salt water: sweat, tears or the sea."
Isak Dinesen

RIP - E4 in the data frame signals to everyone that the sub is dead and the game shall proceed without any further action from that sub.

See figure 2 for an illustration of how a transmission is sent.

Example Transmission

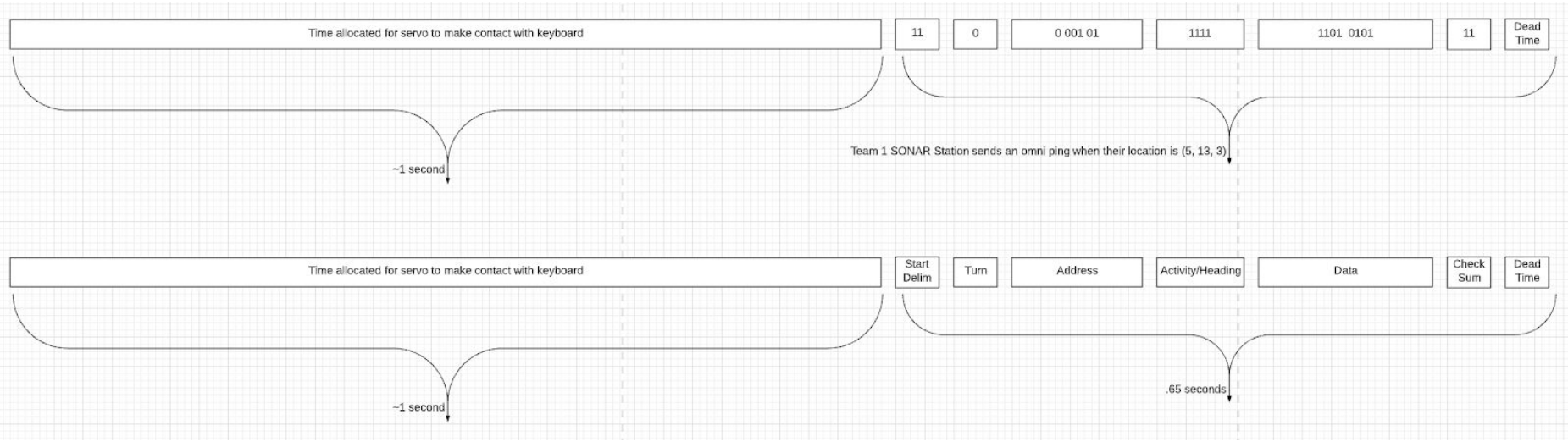


Figure 2. Diagram of how a transmission occurs

"Target? Damned if I can see the horizon!"
--Mike Walsh, U.S.S. Peto

Media Access Control:

Overview:

In order to give all stations a chance to transmit without collision, the protocol defines the use of a time-division multiple access (TDMA) protocol to allow all teams a fair chance to transmit what they need without collision.

Table 5. TDMA frame with the time slots dedicated to each station.

	Team 0			**Team 1**			**Team 2**			**Team 3**			**Team 4**			**Team 5**			**Team 6**			**Team 7**		
Frame	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Tx Time Start (s)	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46
Station	T0 S0	T0 S1	T0 S2	T1 S0	T1 S1	T1 S2	T2 S0	T2 S1	T2 S2	T3 S0	T3 S1	T3 S2	T4 S0	T4 S1	T4 S2	T5 S0	T5 S1	T5 S2	T6 S0	T6 S1	T6 S2	T7 S0	T7 S1	T7 S2

The above table shows the time slots for the TDMA frame. Note that each station will have 2 seconds to send 1 packet of data, but afterwards will need to relinquish the line for the next user. There is currently no mechanism to check for this, so all teams must follow this 2 second window requirement strictly. The start time refers to the transition from the start delimiter to the turn count as this is a clear change in tone.

“There is nothing more enticing, disenchanting, and enslaving than the life at sea ‘or in 218’ ”
--Joseph Conrad, addition by CommComm 2020

Syncing:

At the start of the game, all stations will need to sync up before the game can get underway, with the early bird getting to set the clock. The first station to enter should first listen for an entire TDMA frame cycle to see if there is anyone out there. If it does not receive any signals within said time, it should assume it is the first one online and start transmitting a SYNC activity and should set its turn counter to 0. This will allow other stations to sync up and set their clocks to match. This station should also be on the lookout for new members joining so that it too can sync with that station. The game begins once all stations in a TDMA frame transmit SYNC.

Syncing should be done dynamically, with all stations syncing anytime they receive any information. This will help prevent any relative drift in transition times, as all units will drift together, preventing any bucket collisions. This process works based on the predefined time slicing laid out in table 5. When a station hears a message from another, it shall decipher which station sent the message and then determine how much time after that station's transmission it should transmit. For example, if T4S1 hears a transmission from T6S2, it knows that it needs to wait 18 frame times before it can transmit, and will update its timer telling it when it can transmit. In this way, the stations will constantly be syncing relative to each other every frame, helping to prevent any frame collisions.

If a station goes offline and re-enters, it shall perform this dynamic syncing process over again. It is important to check the turn value to make sure the station is synced with the rest of the units. The protocol for updating the reconnected sub with the new game state information should be determined by the team as with the rest of the internal submarine communication (denoted by a set MSB in the address). The unit should also check for an RIP activity from one of its sister stations to know if it is even alive. Dead stations shall transmit the RIP activity during their frame so other stations can continue to sync.
An important note is that timing is not global - it is relative to T0. Therefore in the syncing process, the philosophy is always "how long do I need to wait before I can transmit something?" Thus, when determining how much time to wait, care has to be taken in how the calculation wraps around.

To walk through an example, T3S1 is the station of interest and will be referred to as "the station". It sits on the line, constantly listening for packets coming by and hears a packet. It will then need to decode who sent the packet. For this example it decodes the packet was sent by T2S1. It then knows from the time slots it needs to wait 6000ms before it can transmit. Another packet comes from T2S2 and now the station re-determines it needs to wait 4000ms to transmit. Then it hears a packet from T3S0 and now knows that it needs to wait 2000ms before it can transmit. In this way, it does not actually matter when T2S2 or T3S0 transmit, as long as it is before 6000ms and 4000ms respectively. Suppose T2S2 actually transmitted when the station thought it had 3800ms to transmit. The station will re-adjust and reset its timer to 4000ms, so that way it will compensate for the 200ms drift caused by T2S2. Since all stations are performing this syncing, all the stations will drift together. See Figure 3 for more details.

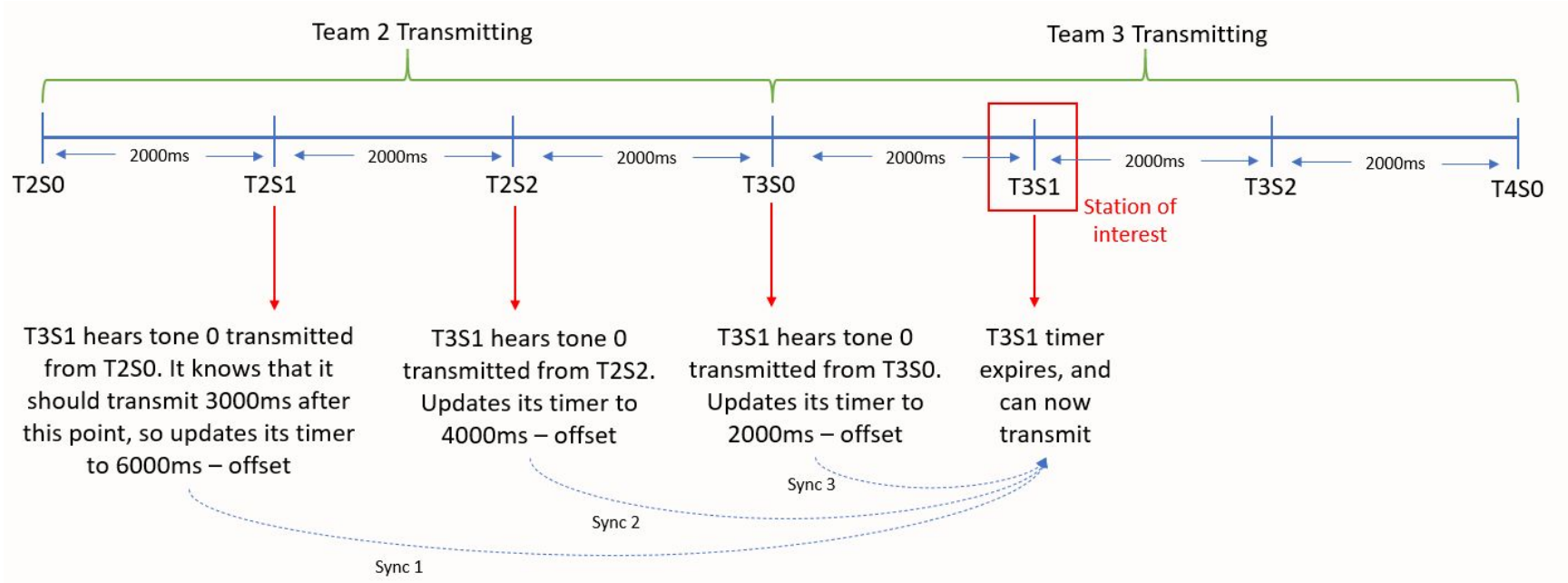


Figure 3. Illustration of dynamic syncing

Another important timing issue to point out is the time it takes before the address information of a packet is sent. Thus, when updating the time to wait for transmit, the amount of time it took from the start of a frame to the determinacy of the address should be considered. From the above example, it would inaccurate to say when the station hears T2S2's packet that it needs to wait for 4000ms from that point, because ~500ms of the message from T2S2 has already passed and therefore the station really only needs to wait ~35000ms before it can transmit. If this is not considered, all stations will drift out by ~500ms and the TDMA sequence will be unnecessarily lengthened by 500ms.

Each station must have some implementation to account for this delay caused by sending messages, reducing the delay to be less than 100 ms between frames. One recommended approach for this is to read the system time when the T0 indicator is heard, and to read the system time after determining who sent the packet, then subtract this time from the frame times. See figure 4 for more details.

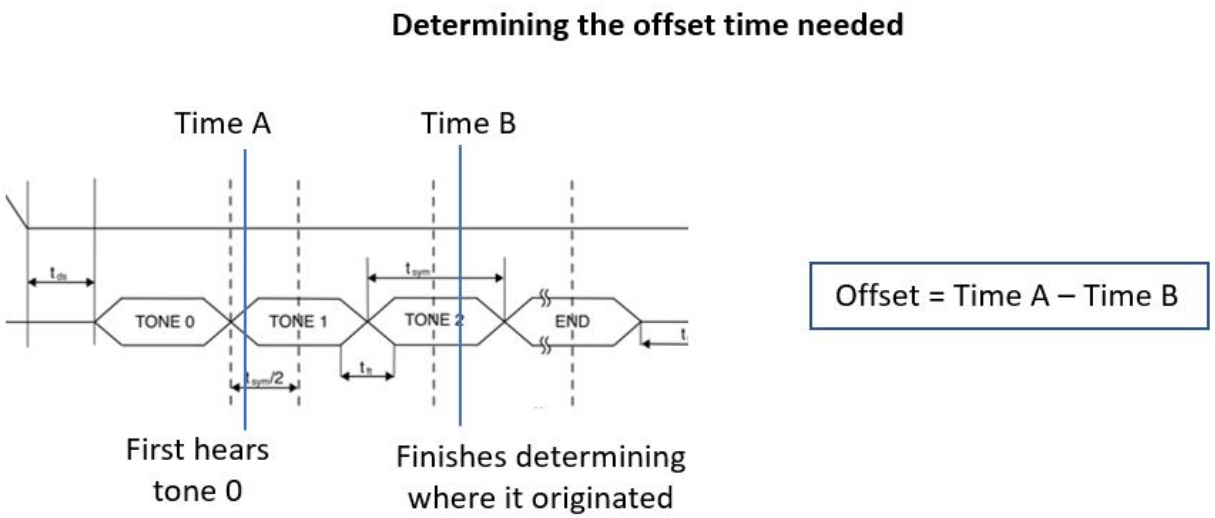


Figure 4. Illustration of offset adjustment

"'Sinking up, how is that even possible?'
-- CommComm 2020"
-- Michael Scott

Determine Station Status (Offline / Online):

The internet is amazing, but for some of us, not the most reliable. There is a very good chance a station's operator will lose internet connection and go dark. When this occurs during the station's transmit time, the line should be quiet for that frame. Since stations are syncing based on the address they recieve, they should be able to tell which station goes dark. Since the order is predetermined in Table 4., When a station hears the address for T1S1, it should expect to hear T1S2 next. If it does not, but hear T2S1 instead, it should know that T1S2 and T2S0 are both down. Knowing which stations are down will be helpful for implementing the Transport layer of the protocol.

Network Layer:

Addressing:

Addressing is done with 6 bits (3 tones) with the most significant bit <5> being the internal/external flag. When the bit is set, this means the transmission is meant for the stations of the source submarine and everyone else should ignore. If the bit is clear, this means the communication is meant for other submarines. The use of this bit is very important, as it differentiates a directional ping from a position update. If the bit is not set when it should be, others will think a directional ping has been sent and you may unintentionally give away your position.

The next 3 bits <4:2> are for the submarine number 0-7. Team 0 will have the number 0, Team 1 the number 1, and so on. The team number can be assigned at compile time or run time as long as it is decided prior to game play, as this is important for syncing and addressing. Finally, the last 2 bits <1:0> are for the station of the source submarine. These values are fixed, see Table 2 for details.

"The sea (or 218), once it casts its spell, holds one in its net of wonder forever."
--Jacques Cousteau, Addition by CommComm 2020

Data flow:

Team Order:

Slot order is always from least to greatest team number and station number (e.g. Team 2 would go before Team 3, and within Team 2, Station 0 would go before Station 1) regardless of the number of teams. The number of teams can be variable, and software must be able to accommodate different team numbers at compile time. For each team not in the main sequence shown in Table 4, decrease the time offsets by three times the station time slot length.

Turns:

There is a global turn counter that all stations need to follow. In each transmission, the turn is represented by a dibit, and is used to ensure all the stations are on the same page about what is going on. When all the stations send a NEXT TURN this means every station is ready for the users to execute actions on the next turn. In NEXT TURN, the data has the 8-bit global turn count. For all transmissions, the stations will all increment the dibit representation of the turn count in a modulus 3 fashion, i.e 0 to 2 then loop back to 0. Since the transitions need to be tracked, having only 1 symbol represent the counter is enough. The turn counter will help when stations drop out to be able to sync back up with game play.

It is possible for a station to send a NEXT_TURN activity during one TDMA cycle, but then send a response on the next cycle, thus if not all stations are sending NEXT_TURN but are still transmitting valid data, the turn cannot advance. However if (a) station(s) goes offline and all active stations are transmitting NEXT_TURN, then an additional cycle shall commence to see if the dropped station(s) are back. If they are still quiet after the additional loop, they are ignored and the turn shall increment. When the dropped stations re-enter the loop, they will be able to sync back using the turn data. If a station is declared dead, transmitting an RIP activity, then it does not need to send a NEXT_TURN since it is no longer part of the game.

After all stations have agreed to increment a turn, each station shall send a NO_ACTION for one complete TDMA cycle to allow operators of a station to determine their actions for the given turn. This will give the teams time to coordinate what they want to do for the next turn based on the information they received in the previous turn. Actions should be entered into the station during this period so said actions can be transmitted on the next TDMA cycle. If no action is input, a default NO_ACTION should be sent and the user should be prevented from entering anything.

“Think or Sink”
--CommComm 2020

Start of Game

At the start of the game, all stations involved will need to sync before the first turn can get underway. Once all stations hear a SYNC from all other stations, they shall put out a NEXT_TURN to allow the game to begin.

In Case of Death

After death, you can still DETONATE your torpedo at the corresponding time. You are still dead and all other messages must be RIP.

Transport Layer:

Error checking of each message is accomplished through the check sum dibit <11> or each data packet, see Table 2. This value is included in every message and can be compared to the calculated checksum through adding each of the data bits and subtracting from 0x3. If these values do not match, there has been an error in sending or receiving that message and the receiving submarine will ignore that message.

"It is not down on any map; true places never are."
--Herman Melville, Moby Dick; Ishmael

Session Layer:

A session is represented by one complete loop of the TDMA frames, with T0S0 starting the session and T7S2 ending it.

Presentation Layer:

There is no encryption or data compression; what you see is what you get. If you want to compress go do it somewhere else.

Application Layer:

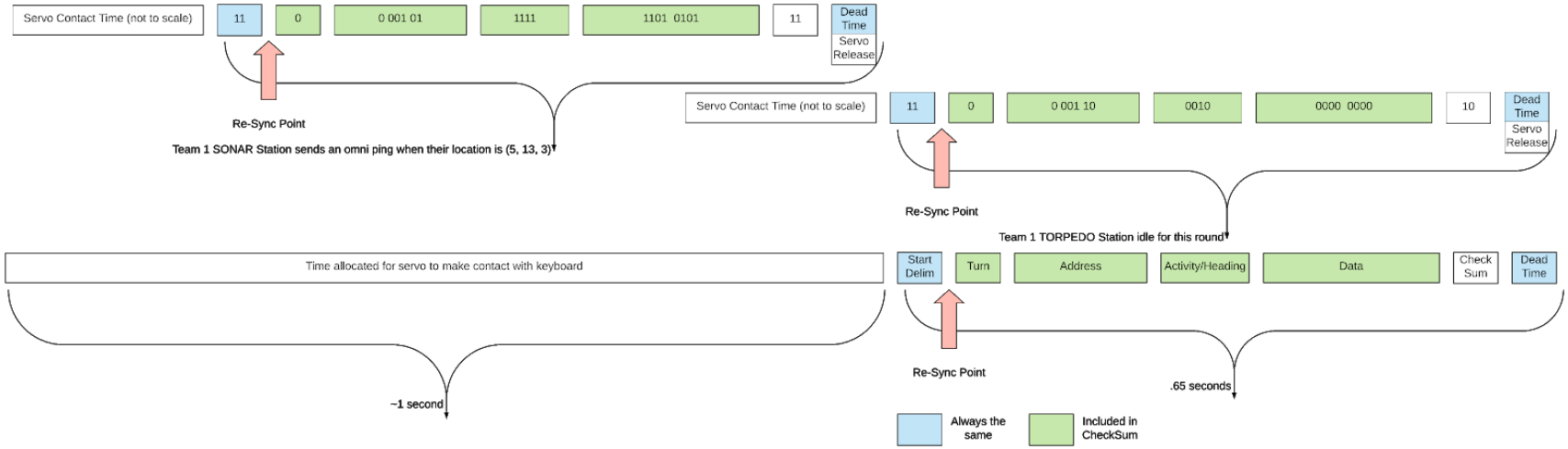
This communication protocol shall be referred to as the Internet of Pings (IOP).

As for me, I am tormented with an everlasting itch for things remote. I love to sail forbidden seas, and land on barbarous coasts.
--Herman Melville, Moby Dick; Ishmael

Appendix

Examples

Example 1: Demonstrates one transition between the same team and shows how checksum was calculated for each of the two transmissions.



CheckSum Calculations

For Team 1 SONAR Station:

Sum:

```
  00
  00
  01
  01
  11
  11
  11
  01
  01
  01
 ---
  10
```

2 % 4 = 2

3 - 2 = 1 = 01

For Team 1 TORPEDO Station:

Sum:

```
  00
  00
  01
  10
  00
  10
  00
  00
  00
  00
 ---
  01
```

1 % 4 = 1

3 - 1 = 2 = 10

Example 2: Demonstrates a couple rounds of actions between 3 teams where T0 has already launched its torpedo at T1 one turn before this (so it detonates in the 2nd round shown below).

Station	T0 CONN	T0 SONAR	T0 TORP	T1 CONN	T1 SONAR	T1 TORP	T2 CONN	T2 SONAR	T2 TORP
Position & Heading	Positioned centrally on the board			Positioned near T0 (within OMNI range)			Positioned along the same heading as T0, within 6 spaces		
1st TDMA	POS_UP DATE	OMNI (will affect T1)	NO_ACT	NO_ACT	NO_ACT	LAUNCH (no effect for 2 turns)	NO_ACT	DIRECT	NO_ACT
2nd TDMA	NEXT	ECHO (only T2 sees)	NEXT	NEXT	ECHO (only T0 sees)	NEXT	NEXT	NEXT	NEXT
3rd TDMA	NEXT	NEXT	NEXT	NEXT	NEXT	NEXT	NEXT	NEXT	NEXT
4th TDMA	HUMAN TIME - If no action is entered, then in the next round, the software should default to transmitting NO_ACTION. NO_ACTION is also transmitting during this time								
Station	T0 CONN	T0 SONAR	T0 TORP	T1 CONN	T1 SONAR	T1 TORP	T2 CONN	T2 SONAR	T2 TORP
Position & Heading	Moved to new position, still central, closer to T1			Same position			Same position		
5th TDMA	NO_ACT	NO_ACT	DETON	NO_ACT	NO_ACT	NO_ACT	NO_ACT	NO_ACT	NO_ACT
6th TDMA	NEXT	NEXT	NEXT	HIT	RIP	RIP	NEXT	NEXT	NEXT
7th TDMA	NEXT	NEXT	NEXT	RIP	RIP	RIP	NEXT	NEXT	NEXT
8th TDMA	HUMAN TIME - If no action is entered, then in the next round, the software should default to transmitting								

	NO_ACTION. NO_ACTION is also transmitting during this time								
Station	T0 CONN	T0 SONAR	T0 TORP	T1 CONN	T1 SONAR	T1 TORP	T2 CONN	T2 SONAR	T2 TORP
Position & Heading	Same position			Same position (dead)			Same position		
9th TDMA	NO_ACT	NO_ACT	NO_ACT	RIP	RIP	DETON (will affect T0)	NO_ACT	NO_ACT	NO_ACT
10th TDMA	HIT	RIP	RIP	RIP	RIP	RIP	NEXT	NEXT	NEXT
11th TDMA	RIP	RIP	RIP	RIP	RIP	RIP	NEXT	NEXT	NEXT

"Life without a defined purpose is similar to a boat without a crew in the middle of the ocean."
--Debasish Mridha

Self-Enforced Rules

- While you will receive information from every team on where they are if you send a ping out, you should only be able to see information about their location if they were within range
- You will need to keep track of where your fired torpedo was supposed to detonate.
- You will need to be able to modify outside of game time your team number and the number of teams playing. Possible ways to accomplish this are easily modifiable constants in code or a configuration state that appears on reset where you can modify the values.
- There shall be no actions during the human time.
- The first loop of TDMA is for actions and the second is for responses. All teams get to perform their action in the first loop and their response to all pending actions in the second round. A team that becomes hit does not have to say so until the second round.
- Keep track of how many teams are still in the game and only celebrate when you are the last man standing

"Sometimes it's easier to catch a fish by throwing a stick of dynamite into the lake"
-- A really bad fisherman