Robots trabajando en Equipo

Alejandro J Malo Tamayo & •MX[0]
Centro de Investigación y de Estudios Avanzados del I.P.N.

***Resumen*— El presente trabajo trata de robots que juegan balompié o fútbol (*football*), bajo las reglas definidas por la *RoboCup*. Los robots se supone conectados a una red inalámbrica por medio de la cual reciben el estado del partido, por medio de mensajes generados por el *Game Controller* (GC), programa controlado por uno de los árbitros. A su vez los robots le comunican su existencia al GC enviándole periódicamente un mensaje. Además la competencia permite la comunicación entre robots, esto se hace en la misma red en donde el GC emite y recibe mensajes de todos los robot participando en el juego. La comunicación entre los robots puede usar también la dirección usada por el GC, la dirección de *broadcast*, esto haría que hasta los robot contrarios recibieran el mensaje. A fin de limitar la comunicación al equipo, en este trabajo se propone el uso de una dirección *multicast* que identifica al equipo. Si bien no todos los equipos implementan la comunicación con el GC o entre robots, su uso tiene la ventaja de que el equipo no es castigados cada reinicio de partido con (1) la remoción de un robot, (2) el posicionamiento de los robots restantes no más adelante de la línea del área de gol y (3) su inmovilización por 15 s después de dada la señal de juego (*PLAY*). El código fuente, en Java™, del GC es público y abierto. Sin embargo, su funcionamiento y uso esta muy poco documentado.**

Palabras clave: Control Distribuido, Control cooperativo, Redes, Comunicaciones, Robótica, RoboCup

I. Introducción

En muchos casos un robot no es suficiente para realizar un trabajo, es necesario un equipo. En un ambiente estructurado, como en una línea de ensamble en una fábrica, los robots pueden tener un control central que los coordina con la línea de ensamble. Esto no ocurre cuando los robot trabajan en ambientes no estructurados o geográficamente muy extensos, como una zona de desastre, es útil que tengan cierta autonomía y que comuniquen sus hallazgos. En este caso puede haber jerarquías fijas o dinámicas y robots especialistas. Este es el caso del presente trabajo que trata del caso de robots que juegan fútbol (*football*). En este caso el ambiente es conocidos, pero no esta completamente controlado. Ya que la posición de la bola y de los "otros" robots en la cancha es dinámica, y las necesita determinar un robot para poder determinar su mejor acción.

El presente trabajo trata de robots que juegan balompié o fútbol, bajo las reglas definidas por (RoboCup, 2012). Los robots se supone conectados a una red inalámbrica por medio de la cual reciben el estado del partido, por medio de mensajes generados por el *Game Controller* (GC), programa controlado por uno de los árbitros. A su vez los robots le comunican su existencia enviándole periódicamente un mensaje. Además la competencia permite la comunicación entre robots, esto se hace en la misma red en donde el GC emite y recibe mensajes de todos los robot participando en el juego. La comunicación entre los robots puede usar también la dirección usada por el GC, la dirección de *broadcast*, esto haría que hasta los robot contrarios también recibieran el mensaje. A fin de limitar la comunicación al equipo, en este trabajo se propone el uso de una dirección *multicast* que identifica al equipo. Si bien no todos los equipos implementan la comunicación con el GC o entre robots, e incluso en campeonatos nacionales no se ha usado, uno esperaría que fuera usada el el campeonato mundial, en el que ha sido aceptado el equipo •MX, su uso tiene la ventaja de que el equipo no es castigado cada reinicio de partido con la remoción de un robot (RoboCup, 2012, 8.2.2), el posicionamiento de los robots restantes no más adelante de la línea del área de gol (RoboCup, 2012, 8.2.4) y finalmente su inmovilización por 15 s después de dada la señal de juego (*PLAY*) (RoboCup, 2012, 8.2.6).

El GC esta escrito en Java™, su código fuente es público y abierto, y puede obtenerse de `http://sourceforge.net/projects/robocupgc/develop`. Aunque su funcionamiento y uso esta muy poco documentado, el GC se usa en todas las ligas de humanoides de la RoboCup: kidsize, teensize, adult size y plataforma standard.

La idea de robots jugando fútbol fue mencionada por primera vez por el profesor Alan Mackworth (University of British Columbia, Canada) en su trabajo *"On Seeing Robots"* (Mackworth, 1993) presentado en la conferencia *Vision Interface'92*. El profesor Mackworth veía al fútbol como un campo lo suficientemente difícil como para impulsar la investigación, pero lo suficientemente limitado como para permitir obtener resultados. Su equipo publicó varios trabajos sobre el proyecto Dynamo (Barman *et al.*, 1993; Sahota y Mackworth, 1994). Meses más tarde investigadores Japoneses organizaron un taller sobre Grandes Retos en Inteligencia Artificial (October 1992) en Tokyo. Las discusiones concluyeron con el lanzamiento de la *Robot J-League*[1], una competencia de fútbol entre equipos de robots Esta iniciativa fue abierta a otras naciones, a solicitud de la comunidad internacional ha evolucionado a lo que es hoy la RoboCup que desde 1997 se realiza anualmente, este año

[0] en el orden alfabético de las instituciones que participan •MX incluye a Juan Manuel Ibarra Zannatha, Manuel Heladio Hunter Sánchez, Eric Hernández Castillo, Jorge Enrique Lavín Delgado, Ángel David Gómez Sánchez del Cinvestav, Adalberto Hernández LLarena, Abner Quiroz Clemente, Ignacio López Peña, Lauro Fernando Vázquez Alberto, Héctor Rodrigo Arce González, Eduardo Ruíz Libreros, Marcos Abner, Ramos Ramírez, Ricardo Vargas Morales, Boris Escalante Ramírez de la UNAM y Felipe de Jesús Khamal Lara Leyva de Verstand Labs

[1] **J-League** es el hombre de la liga Japonesa de fútbol profesional

se realiza en México.

El presente trabajo esta organizado como sigue. Primero se describe el contexto donde ser realiza la comunicación. Después se describe y analiza el contenido de los mensajes intercambiados entre el *Game Controller* y los robots. A continuación el contenido propuesto para el mensaje transmitido entre los robots del equipo.

II. Redes de computadoras

La comunicación entre computadoras inicialmente uso los medios de comunicación existentes basados en circuitos conmutados, e.g. las teleimpresoras, sucesores del telégrafo (aún usadas por la industria aeronáutica) y modems en la red telefónica de voz. El uso redes basadas en circuitos conmutados a dado lugar hoy a redes basadas en paquetes conmutados, tecnología sobre la cual se basa *Internet* y la *World Wide Web (W3)*.

	Modelo OSI	Familia de Protocolos Internet
7	aplicación	aplicación
6	presentación	
5	sesión	
4	transporte	TCP **UDP**
3	red	**IPv4**, IPv6
2	enlace de datos	Ethernet (IEEE 802.3)
1	físico	**Wi-Fi (IEEE 802.11)**

Figura 1. Comparación de protocolos OSI e Internet, en negritas los usados en el proyecto

En un inicio, en redes de computadoras, existían una multitud de protocolos, dentro de los cuales destacan SNA de IBM, OSI de ISO, NetBios de Microsoft, XNS de Xerox, MAP/TOP de General Motors. Sin embargo, la familia de protocolos Internet, cuya investigación y desarrollo fue dirigido por la Agencia de la Defensa de Proyectos Avanzados de Investigación (DARPA), los ha desplazado. La familia de protocolos Internet, conocida como TCP/IP, define cuatro capas (Braden (ed.), 1989): Link, *Internet*, *Transport* y *Application*, como muestra la figura 1. En la capa de enlace (*Link*) el uso de Ethernet (Metcalfe y Boggs, 1976) (IEEE 802.3), desarrollado inicialmente por Xerox PARC, y posteriormente por DEC, Intel y Xerox, es común, ya que ha sido implementado en una multitud de medios físicos: cable coaxial, fibra óptica, par trenzado, e inalámbrico. En este contexto se usa una red inalámbrica IEEE 802.11 en la capa de enlace, la versión 4 del protocolo de internet (IPv4) en la capa de internet y UDP en la capa de transporte. La aplicación usa la interfaz de programación de aplicaciones (API) para protocolos de internet, i.e. sockets de red, para implementar la comunicación.

IPv4 utiliza direcciones de 32 bits o 4 bytes. El uso de IPv4 en una red inalámbrica local privada con un número limitado de dispositivos no es un inconveniente.

En una red como internet existen esencialmente dos tipos de protocolos en la capa de transporte. Protocolos en donde los dispositivos realizan una conexión, como *Transmission Control Protocol* TCP y protocolos donde no hay una conexión como *User datagram protocol* UDP. El primero TCP garantiza la recepción de la información, pero esta actualmente restringido a comunicar dos dispositivos. El segundo, UDP no garantiza la recepción, sin embargo, cuando se usa la dirección de *broadcast* todas las computadoras de la red lo reciben. El procesamiento del mensaje depende de si existe un programa escuchando en el puerto. Una variante es el *multicast* en donde sólo un grupo de las computadoras conectadas a la red lo reciben. RoboCup usa *broadcast* UDP para la comunicación con el árbitro.

III. La red inalámbrica de RoboCup

Las redes inalámbricas en interiores son ubicuas hoy en día, en exteriores no lo son tanto. Aunque quizás sean socialmente más importantes, por ejemplo la Red Caza Terremotos (*The Quake-Catcher Network*) con más de 1200 sensores sísmicos usada para la alerta temprana de temblores, algunos colocados en la costa del Pacífico Mexicano y abierta a la participación pública, con integración de investigadores de Estados Unidos de América, México, Colombia, Chile. Pero su uso no se restringe a terremotos, existen aplicaciones en áreas como la agricultura, medio ambiente, meteorología, control de fronteras, ambientes hostiles, fuegos forestales, monitoreo de signos vitales, etc. generalmente usando energía solar.

Si bien Robocup es una competencia que auspicia cooperación entre robots, no es la única. Un ejemplo del interés en el área fue la competencia la *Multi Autonomous Ground-robotic International Challenge* (MAGIC) organizada por la *Tank Automotive Research, Development and Engineering Center* (TARDEC) y la *Defense Science and Technology Organization* (DSTO) en 2010 en Australia con un premio de 1.6 millones de dolares. Robocup no otorga premios en dinero.

RoboCup establece en (RoboCup, 2012, 4.5) el uso de una WLAN (wireless local área network), un límite de 1Mbit/s de ancho de banda por equipo y comunicación de datagramas sobre internet IP/UDP (*user datagram protocol*). La WLAN la suponemos alguna variante de Wi-Fi IEEE 802.11a/b/g/n (IEEE, 2007) ya que el reglamento no especifica una específica; generalmente los ruteadores manejan un subconjunto de las variantes del 802.11.

RoboCup en IPv4 (Protocolo de Internet versión 4) usa la dirección 255.255.255.255.255 para referirse a la red local como dirección de *broadcast*, IPv4 permite tener broadcast en una subred más grande. Para *multicast* IPv4 usa las direcciones que van de 224.0.0.0 a 239.255.255.255, algunas están reservadas.

El contenido del mensaje del *Game Controller* es mostrado en en listado 1. Cada paquete contiene la información contenida en la estructura `RoboCupControlData`. El estado del partido es codificado en las primeras 11 líneas, en la línea 12 se incluye la estructura `TeamInfo` dos veces, con información correspondiente a los equipos, el mensaje supone la existencia de 11 jugadores, en la práctica se tiene autorizado un máximo de 5 jugadores por equipo, donde tres pueden estar en la cancha.

```
1  struct RoboCupGameControlData {
2      char    header[4];           //
           header to identify the
           structure
3      uint32 version;              //
           version of the data structure
4      uint8 playersPerTeam;        //
           The number of players on a
           team
5      uint8 state;                 //
           state of the game (STATE_READY
           , STATE_PLAYING, etc)
6      uint8 firstHalf;             // 1
           = game in first half, 0
           otherwise
7      uint8 kickOffTeam;           //
           the next team to kick off
8      uint8 secondaryState;        //
           Extra state information - (
           STATE2_NORMAL,
           STATE2_PENALTYSHOOT, etc)
9      uint8 dropInTeam;            //
           team that caused last drop in
10     uint16 dropInTime;           //
           number of seconds passed since
            the last drop in. -1 before
           first dropin
11     uint32 secsRemaining;        //
           estimate of number of seconds
           remaining in the half
12     struct TeamInfo teams[2];
13 };
```

Listado 1. Estructura RoboCupGameControlData

La estructura `TeamInfo`, listado 2, contiene información sobre los equipos; alguna invariante: línea 2: el número del equipo no cambia durante el campeonato), otra varía de un partido a otro, línea 3: el color del equipo e información que varía de un tiempo a otro, e.g. línea 4: el color de la portería. O de un instante a otro como el marcador en la línea 5. Además contiene la estructura `RobotInfo` con información de once jugadores.

```
1  struct TeamInfo {
2      uint8 teamNumber;            //
           unique team number
3      uint8 teamColour;            //
           colour of the team
4      uint8 goalColour;            //
           colour of the goal
5      uint8 score;                 //
           team's score
6      struct RobotInfo players[
           MAX_NUM_PLAYERS];        // the
            team's players
7  };
```

Listado 2. Estructura TeamInfo

La estructura `RobotInfo`, listado 3 contiene información sobre cada uno de los robots que participan en el juego. Línea 2 si esta penalizado y línea 3, cuanto tiempo falta para que la penalización termine.

```
1  struct RobotInfo {
2      uint16 penalty;              //
           penalty state of the player
3      uint16 secsTillUnpenalised; //
           estimate of time till
           unpenalised
4  };
```

Listado 3. Estructura RobotInfo

A su vez cada robot envía un mensaje al `GameController`, cuyo contenido consiste de una estructura `RoboCupGameControlReturnData`, listado 4, que en el caso de la liga *kid size* se limita a enviar siempre el mismo mensaje, equivalente a decir que el robot esta vivo (`message=GAMECONTROLLER_RETURN_MSG_ALIVE=2`), línea 6 del listado.

```
1  struct RoboCupGameControlReturnData {
2      char    header[4];
3      uint32  version;
4      uint16  team;
5      uint16  player;              //
           player number - 1 based
6      uint32  message;
7  };
```

Listado 4. Estructura RoboCupGameControlReturnData

En al interfaz del GC, mostrada en la figura 2, el testigo de un jugador se ilumina de color verde si el último mensaje fue recibido hace menos de 1,500 ms; rojo si el último mensaje fue recibido hace más de 2,500 ms; y amarillo si el último mensaje fue recibido hace más de 1,500 ms y menos de 2,500 ms. Esto es mostrado en la figura 1, donde cada jugador es representado por un rectángulo con su número (tercera y cuarta columnas a partir de la izquierda), el testigo ocupa la esquina superior derecha de cada jugador.

Considerando que el robot envía un mensaje cada cada 500 ms; dado que cada mensaje tiene 16 bytes de longitud (128 bits) y hay hasta un máximo de 5 robots por equipo. Considerando (Postel (Ed.), 1980; Postel (Ed.), 1981) un encapsulamiento debido a UDP de 8 bytes y a IPv4 de 20 bytes, se tiene 352 bits por mensaje, se tienen 3,520 bits/s

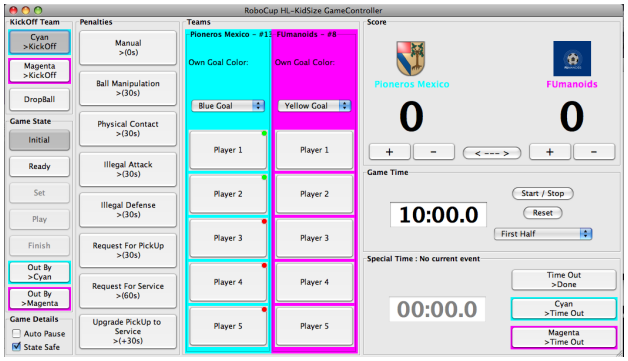

Figura 2. Interfaz del *Game Controller*, programa usado por los árbitros. Los puntos verdes en la esquina de los jugadores indican que el programa recibe mensajes de los robot 1 y 2 del equipo cían.

ocupados por los mensajes que envían los robots al GC, quedando 996,480 bits/s para mensajes entre los robots del equipo. En algunos casos UDP restringe la longitud del paquete a 512 bytes. UDP supone el mensaje completo en un paquete.

IV. Módulo de Comunicación

El programa se diseño con cuatro hilos, dos que reciben mensajes y dos que envían mensajes. De los hilos que reciben mensajes, uno recibe el mensaje del árbitro, y otro el del grupo. De los hilos que envían mensajes uno envía un mensaje al árbitro, para indicar que esta vivo; y otro envía el mensaje a los miembros del equipo.

La elección del uso de hilos se hizo debido a la libertad que ofrecen para activar y desactivar su ejecución. Lo que permite el envío y recepción de mensajes en forma periódica, consumiendo pocos recursos.

La figura 3 muestra el sistema operativo del robot en *statecharts*, el módulo de comunicación es parte de éste. La figura muestra, a grandes rasgos los principales componentes y como éstas reaccionan a su entorno, se omite la comunicación entre hilos y los eventos que disparan los cambios de estado. Existen programas que generan código en forma automática a partir de este tipo de figura, e.g. Statemate(Harel y Politi, 1998), Esta gráficas son equivalentes a redes de Petri jerárquicas coloreadas (Jensen, 1996) y han sido integradas al *Unified Modeling Language (UML)* usado en la programación orientada a objetos.

IV-A. Broadcast con el árbitro

El árbitro usa un programa para comunicarse con los robots participando en el juego, la interfaz es mostrada en la figura 2. la información que reciben los robots es mostrada en la figura 4. En la primera línea vemos que el robot recibo un mensaje del árbitro de 116 bytes y esta ha sido el mensaje 169 recibido. En la segunda línea vemos que el robot envía al GC un mensaje de 16 bytes, este es el mensaje que le dice al GC que el robot esta vivo. A partir de la quinta línea se despliega el contenido del mensaje, primero información sobre el tipo de mensaje y después sobre el estado del juego, finalmente se lista la información de los dos equipos. Un equipo puede tener un máximo de cinco jugadores.

El mecanismo que se usa en este intercambio entre un robot y el árbitro es el de *broadcast*, por lo que todos los robots y las computadoras conectadas a la subred lo reciben, incluso los mensajes el mismo robot envía. El usar el árbitro electrónico tiene como ventaja de que el equipo no es penalizado durante los arranques del juego.

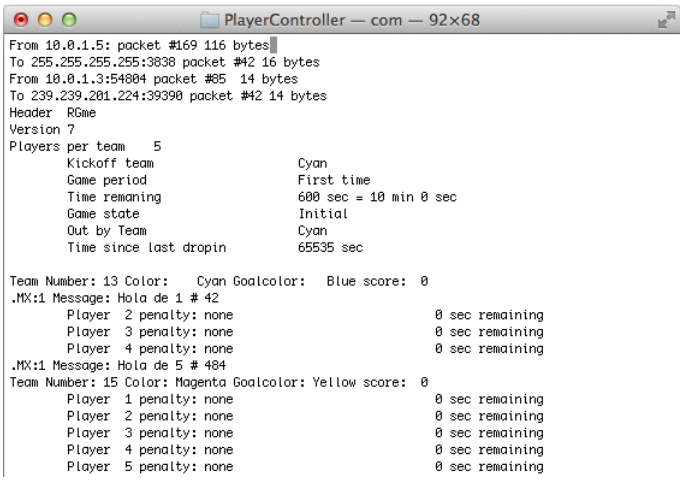

Figura 4. Despliegue de la información recibida del GC y de los robots del equipo

IV-B. Multicast del equipo

En la misma figura 4 se muestra el diálogo entre los robots del equipo que se realiza usando *multicast*, el grupo tiene la dirección 239.239.201.224 y usa el puerto 39390. En la tercera línea se puede observar que se ha recibido el mensaje 85 del robot con dirección 10.0.1.3 por el puerto 54804 de 14 bytes, el contenido de este mensaje es mínimo sólo contiene el número de mensaje y el jugador que lo envía. Al mismo tiempo se puede ver en la parte baja que hay dos robots activos, el 1 y el 5, una parte mínima del mensaje es sobrescrita en la línea en donde se escribe la información que envía el GC sobre el jugador.

Los jugadores de un equipo pueden comunicarse, respetando el ancho de banda asignado. La comunicación entre los robots del equipo se realiza usando *multicasting*, en este caso el mensaje es transmitido a los robots y a las computadoras conectadas al grupo, lo que permite limitar el acceso al mensaje

El mensaje esta definido por una estructura `TeamMsg`, listado 5, que contiene el número del jugador que envía el mensaje (línea 3), el instante en que fue enviado (línea 4), el estado del robot (línea 5) definido por una estructura `State` y los objetos que observa, definidos por una estructura `Object` (línea 6)

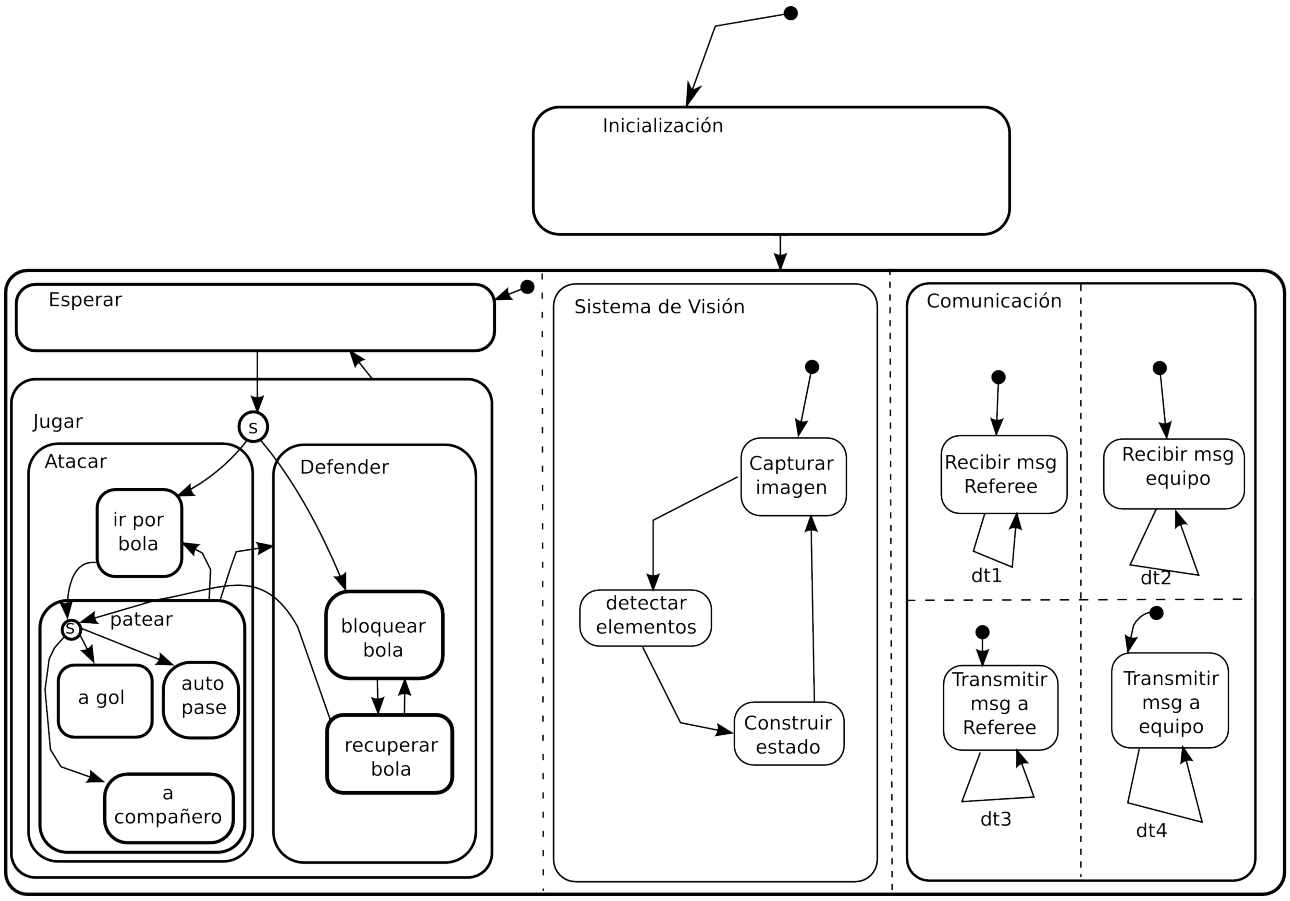

Figura 3. *Statechart* de Harel describe al software del robot como un sistema reactivo.

```
1  struct TeamMsg {
2          char[3] header;      // ".MX"
3          uint8 player;        //
               número del jugador
4          unit32 time;         // hora
               del mensaje
5          struct State;        //
               estado del robot
6          struct Object[11];   //
               objetos observados
7  }
```

Listado 5. Estructura TeamMsg

```
1  struct State {
2          uint8 position; // 0 portero,
               1 defensa, 2 ofensivo
3          uint8 state;    // 0 OK, 1
               Derribado, 2 fuera del
               campo
4          uint8 x;        // posición en
               el campo, x transversal,
               y longitudinal
5          uint8 y;        //
               {[-200,200],[-300,300]}
6          uint8 theta;    // dirección
               (-pi, pi]
7          uint8 battery; // [0-100%]
8  }
```

Listado 6. Estructura State

El estado del robot es codificado en una estructura `State` la cual contiene la posición que juega el robot (línea 3), el estado actual (línea 4), la posición en el campo (líneas 4 y 5), relevante sólo si el robot esta OK, la dirección en la cual apunta (línea 6) y la carga de la batería (línea 7). Esto facilita a los otros robots del equipo decidir si pueden considerar al robot como jugador activo de la jugada.

El estado de todos los objetos "dentro" de la cancha son codificados en la estructura `Object`, en este caso se indica el número de objetos (2), el tipo de objeto (3), la dirección relativa (4) y la distancia aproximada (5), en caso de disponer de ésta información.

```
struct Object {
        uint8 number;    // número
            total de objetos
            observados
        uint8 item; // 0 bola, 1
            robot del equipo, 2 robot
            contrario, 30 portería B
            poste izq, 31 portería B
            poste der, 40 portería Y
            poste izq. 41 portería Y
            poste der. 5, marca BYB 6,
             marca YBY 7
        uint8 bearing; // -180,180
            ángulo relativo [left-
            right]
        uint8 distance; // 0-720
            distancia
}
```

Listado 7. Estructura Object

La distancia al balón es un parámetro usado para determinar quien va por él, y en que plan, ofensivo, si es un jugador del equipo o defensivo, si lo es un jugador del otro equipo.

V. Conclusiones y Trabajo Futuro

En Robocup si los robot tiene un rol fijo, la comunicación no es necesaria. Si se tienen descomposturas, castigos durante el juego, un esquema rígido pone al equipo en desventaja. Por otro lado, si se desea un equipo flexible, con cierto grado de conciencia de lo que hace el compañero, la comunicación es útil.

La información disponible por un robot le puede permitir asumir el rol de líder en una jugada, permitiendo que varios robots trabajen como uno.

En este trabajo se presenta un esquema en donde el mensaje tiene una estructura fija. Sin embargo, esta es una limitación que se puede evitar. El uso de mensajes de contenido variable es perfectamente posible. La única desventaja radica en el mayor procesamiento necesario.

Los mensajes no están encriptados, en ese sentido cualquiera los podría interceptar. Sin embargo, sin un entendimiento del contenido es difícil que puedan ser usados en contra por los contrincantes. Al multicasting se podría se le podría agregar el encriptamiento o algún mecanismo de verificación. Es claro que si el contenido de un mensaje no cumple con el formato debe ser desechado.

En este trabajo se ha resuelto ha desarrollado una plataforma que permite la comunicación, su explotación es objeto de trabajo posterior. El área de control distribuido en red es actualmente objeto de investigación, la cooperación abre la puerta a la detección y seguimiento de objetos móviles (DATMO) en forma cooperativa.(Bullo *et al.*, 2009; Hristu-Varsakelis y Levine, 2005; Saligrama, 2008; Wang y Liu, 2008). Otro aspecto a considerar es la aplicación de la teoría de juegos en el control de los robots(Isaacs, 1967; Başar y Olsder, 1999; Papadimitriou, 2001) Nuestro trabajo en este aspecto apenas comienza.

Referencias

Barman, R., S. Kingdon, J.J. Little, Alan K. Markworth, D.K. Pai, Michael K. Sahota, H. Wilkinson y Y. Zhang (1993). DYNAMO: real-time experiments with multiple mobile robots. En: *Proceedings of the Intelligent Vehicles Symposium*. pp. 261–266.

Başar, Tamer y Geert Jan Olsder (1999). *Dynamic Noncooperative Game Theory*. second ed. SIAM.

Braden (ed.), Robert (1989). Requirements for internet hosts – communication layers. Reporte técnico . Internet Engineering Task Force.

Bullo, Francesco, Jorge Cortés y Sonia Martínez (2009). *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Applied Mathematics Series. Princeton University Press.

Harel, David y Michal Politi (1998). *Modeling Reactive Systems with Statecharts: The Statemate Approach*. McGraw Hill.

Hristu-Varsakelis, Dimitrios y Levine, William S., Eds. (2005). *Handbook of Networked and Embedded Control Systems*. Birkhäuser Boston.

IEEE (2007). Std 802.11. Reporte técnico . The Institute of Electrical and Electronic Engineers, Inc.

Isaacs, Rufus (1967). *Differential Games: a mathematical theory with applications to warfare and pursuit, control and optimization*. SIAM series in Applied Mathematics. John Wiley & Sons, Inc.

Jensen, Kurt (1996). *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Vol. 1 de *Monographs in Theoretical Computer Science*. 2 ed. Springer Verlag. Berlin,.

Mackworth, Alan K. (1993). On seeing robots. En: *Computer Vision: Systems, Theory and Applications* (A. Basu y X. Li, Eds.). pp. 1–13. World Scientific Press. Singapore. Reprinted in P. Thagard (ed.), *Mind Readings*, MIT Press, 1998.

Metcalfe, Robert M. y David Boggs (1976). Ethernet: distributed packet switching for local computer networks. *Communications of the ACM* **19**(7), 395–404.

Papadimitriou, Christos H. (2001). Algorithms, games, and the internet. En: *In Proceeding of the thirty-third annual ACM symposium on Theory of Computing*. Hersonissos, Crete, Greece. pp. 749–753.

Postel (Ed.), Jon (1980). User datagram protocol. RFC 768. Defense Advanced Research Projects Agency. Information Sciences Institute, University of California.

Postel (Ed.), Jon (1981). Internet protocol. RFC 791. Defense Advanced Research Projects Agency. Information Sciences Institute, University of California.

RoboCup (2012). *RoboCup Soccer Humanoid League Rules and Setup (for the 2012 competition in Mexico City)*. final version of June 6, 2012 ed. RoboCup Humanoid League. http://www.tzi.de/humanoid/.

Sahota, Michael K. y Alan K. Mackworth (1994). Can situated robots play soccer?. En: *Proceedings of the Artificial Intelligence '94*. pp. 249–254.

Saligrama, Venkatesh, Ed. (2008). *Networked Sensing Information and Control*. Springer-Science+Business.

Wang, Fei-Yue y Liu, Derong, Eds. (2008). *Networked Control Systems: Theory and Applications*. Springer Verlag London Ltd.