Wireless Control of Mobile Robot Squad with Link Failure

Mikael Pohjola
Department of Automation and Systems Technology
Helsinki University of Technology
Espoo, Finland
mikael.pohjola@tkk.fi

Shekar Nethi, Riku Jäntti
Department of Communications and Networking
Helsinki University of Technology
Espoo, Finland
{shekar.nethi, riku.jantti}@tkk.fi

Abstract—**This paper deals with control during breaking communications. We specifically examine the breaks of communication due to mobility in a wireless multihop robot control system. We compare the applicability of singlepath and multipath routing protocols without and with priorization for control of mobile robots. We investigate the performance and recovery of a wireless network with high mobility. The issue is also looked at from the control point, where two packet-loss-robust control methods are employed. The issue of control during link break with long recovery is considered. The matters are demonstrated with a joint control and communication simulator, PiccSIM, as a scenario of a mobile robot squad is demonstrated.**

Keywords: wireless multihop mobile robot, dynamic position, link break, priorization, rerouting

I. Introduction

The control of mobile robot squads is a very challenging task. While each robot could localize itself and control its movement autonomously, collaboration among the robots requires them to exchange information – typically using wireless communications. If the collaborating robots are spread over large area, then multihop communications are needed in order to allow any-to-any communication. The wireless network formed by the robots is an example of a mobile ad hoc network which is characterized by dynamically changing network topology. While the robots move, existing radio links are broken and new links formed. This imposes challenge to the coordination of the robots since 100% packet delivery cannot be assured.

In this paper, we consider the special case, in which the actions of the robots are controlled by a single squad leader. The leader collects the state information from the robots and then transmits movement commands to them in order to coordinate the movement of the overall robot squad such that the desired formation of the robots could be achieved. In this setting, there is need for co-design of control and communications, since the ability of the robot squad to follow a trajectory decided by the leader depends both on the utilized control strategy as well as the delays and packet losses caused by the utilized communications strategy. A control loop can cope with heavy packet loss and there are theorems that still guarantee stability of the control loop [4]. But if consecutive packet losses are long, control becomes difficult. Typically long communication delay leads to an outage situation where robots out of the control of the leader must be stopped and their control halted. In the squad operation, the control performance depends on the performance of all the robots. Hence, the message flow between the leader and all the robots are of equal importance regardless of the number of hops between them. In communications design, this means that some form of preemption should be used based on the number of hops packets need to travel. In [8] it was argued that by using a routing protocol that sets up multiple paths between source and destination can significantly increase the robustness of wireless control system by reducing the packet loss and jitter. This conclusion was drawn by observing a system where fixed relays were utilized in the communication between controller and moving robot. Our results here, however, indicate that multipath routing is not beneficial if all the nodes are moving, since all the paths to the destination are very likely to break (almost) simultaneously.

PiccSIM is a joint network and control simulation tool developed at Helsinki University of Technology. In this paper the capabilities of PiccSIM is extended by a dynamic position update mechanism. The position of mobile nodes can be updated from the simulation model to the Ns2 simulator. This enables the simulation of any kind of mobile robot scenarios. The technical details of the dynamic position update are explained in Section II.C. PiccSIM is used in this paper to gain more understanding of control of mobile robots with route breaks, causing long outages during which control of the robots is impossible. A scenario with a robot squad which changes formations facilitates as a case example. The change of formations causes the communication paths to break.

To our knowledge there is only one other simulation tool that provides simulation of mobile robots with dynamic position update in the Ns2 simulator [14]. They use Areena, which is a dedicated robot simulation environment, integrated with the Ns2 simulator for inter robot communication. The synchronization of the positions in Areena and Ns2 is handled with a TCP connection between the two simulators. PiccSIM is more general because it allows simulation of any dynamic system with mobile wireless nodes, not just robots.

This paper is organized as follows: The PiccSIM platform and the extension developed for dynamic node position update is described in more detail in Section II. Then control with

WIOPT 2008, 1st – 3rd Apr 2008, Berlin, Germany.
Copyright © 2011 – 2012 ICST ISBN 978-963-9799-18-9
DOI 10.4108/ICST.WIOPT2008.3126

breaking communications is discussed, including data priorization, in Section III. The results are demonstrated with simulations of a mobile robot squad with multihop control communication in Section IV. Finally Section V concludes the paper.

II. The PiccSIM Platform

PiccSIM stands for "Platform for integrated communications and control design, simulation, implementation and modeling" and as the name suggests offers a joint simulation platform for NCS, where both the network and the control system is simulated. PiccSIM is based on the MoCoNet platform designed for remote laboratory experiments [12].

The PiccSIM platform was developed to support simulation of wireless automation cases both for teaching and research. The new addition that we present in this paper is the dynamic position update mechanism, allowing simulation of any mobile node or robot scenario.

PiccSIM consists of a control design and simulation model done in MATLAB/Simulink and the network simulation done in Ns2. MATLAB and Simulink are widely employed research tools used in dynamical system simulation, and control engineers are accustomed to them. Ns2 on the other hand is the de facto standard for network simulation in the communication research community [10]. Models for new wireless technologies, such as routing protocols, are frequently developed for Ns2. By combining the strengths of both tools, a flexible and efficient joint simulation platform is obtained. This approach not only offers all functionalities of both simulators but also provides modularity for researchers developing on either system. This integration has previously been reported and we refer to [9] and [8]for more technical details.

There are also other comparable simulation tools available. The most relevant tool to PiccSIM appears to be Modelica + ns2 [2]. It is a very similar platform developed at Case Western Reserve University (USA). Alike as PiccSIM, the plant dynamics and the control simulation is done in Modelica and the network simulation in Ns2. In this implementation both components exchange information with each other to synchronize the simulation of the system in both control and network domains. The simulation is controlled by Ns2, and Modelica is instructed to run until a certain time, upon data synchronization is performed. Then Ns2 simulates until the next read or write from the dynamic simulation in Modelica is performed. The communication traffic is, though, generated by Ns2 and specified in advance. No event based (e.g depending on unforeseen events in the dynamic model) communication can be done. The packet rates, sources and destinations are specified in the tcl script for Ns2 before simulation. PiccSIM, on the other hand, generates its traffic in the dynamic simulation model and event based sending of packets is possible.

In the next subsections an overview of the PiccSIM architecture, the network simulation and the new position update extension is given. In the simulations of Chapter IV a case scenarios is introduced, where the new capabilities of PiccSIM are presented.

A. *Overview*

The architecture of PiccSIM is depicted in Fig. 1. The PiccSIM system typically consists of three computers (a server, xPC Target and the Ns2 computers) and an I/O board (connected to the real process). These are attached by a LAN to a gateway, such that users on the Internet can connect to the system and operate it remotely. The user interface is implemented as a Java applet.

The server computer is responsible for the user connections. The xPC Target computer runs a real-time operating system and controls the real process or simulates a process in real-time. Simulink models built by the user describing the control system are automatically compiled using the automated code generation capabilities of MATLAB to executable code (rapid control prototyping) and uploaded to the xPC Target where it is run.

The network of the wireless control system is simulated by the Ns2 computer. For networked control systems simulations the communication is done with UDP packets. Since a control system do not need reliable transfer of packet, but a lightweight container for a small amount of data, UDP suits better than TCP. Packets sent over the simulated network are routed through the Ns2 computer, which simulates the network according to any tcl script specification.

The communication over the simulated network is handled by a readymade library of blocks shown in Fig. 2. The user need not pay attention to the integration of Simulink and Ns2. The source and destination IDs of the communicating nodes, the data types and dimensions of the signals to send need to be configured. The conversion to UDP packets and back to Simulink signals is handled by the network node blocks.

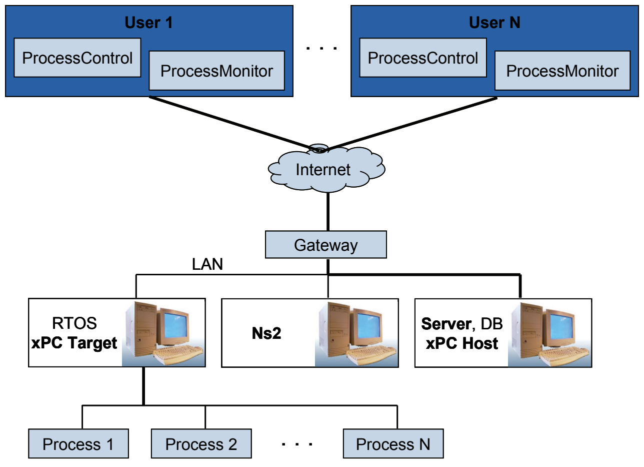

Figure 1. Architecture of the PiccSIM platform.

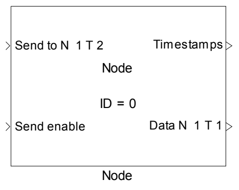

Figure 2. Simulink block for communicating over a network. Different types (T) of packets can be sent/received to/from different nodes (N).

B. *Network Simulation*

The network simulator uses the emulation mode of Ns2 known as NSE (Network Simulator Emulator). The network simulator is running on SUSE 9.3 and has Ns2.30 installed on it. The capabilities of the current Ns2 version, has been extended to suit the requirements of the PiccSIM system.

The Ns2 computer captures the UDP packets from the LAN (with so called taps) and injects them into the simulated (wireless) network model. The network is simulated in real-time with the NSE extension, with packet drops and delays. Once the packet has reached the destination in the simulated network it is sent back to the xPC Target. Thus communication between any two nodes in the control system is passed through the simulated network of Ns2. Fig. 3 shows the connectivity mapping between the xPC Target and the Ns2 nodes. UDP port numbers are used to associate packets to the corresponding node in Ns2. In the figure, Node 1 is the source and Node 5 the destination. Node1 on the xPC Target represents e.g. a sensor. It creates UDP packets of the signals measured from the process and sends it to the Ns2 computer. The Ns2 computer captures the UDP packets (with a Tap) and associates it to the correct node in the simulated network by the UDP destination port number. The packet transmission is simulated in Ns2. On successful reception of the packet inside the simulated network, the packet is sent back to the xPC Target via the LAN. The corresponding Simulink UDP receive block captures the packet, converts it to a Simulink signal and outputs it to the rest of the simulation model.

C. *Dynamic position extension*

In many robotic scenarios mobility is essential. A classic example is a search and rescue operation. Consider a group of mobile robots scattered in an area. The task is to perform an efficient sweep or search of the area. This requires negotiating between the robots, to divide the task between the robots to cover the search area efficiently. Communication links must be established between the robots and maintained throughout the operation. This includes rerouting, when the robot formation changes or obstacles shadow the radio signal, during the search. One leader node could co-ordinate the movements of the sensing robots. For robot squads of three or more, the network will be a multihop network as the robots spread out in the field, past the radio range.

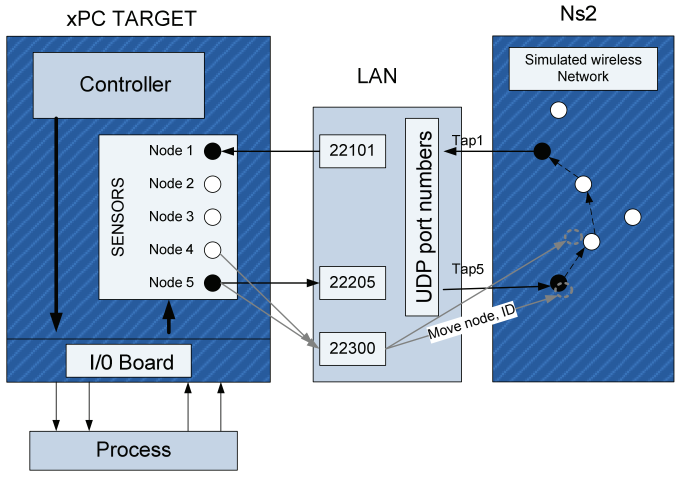

Figure 3. Communicating over Ns2 simulated network. Dynamic position update indicated with gray.

Other applications such as RoboSoccer also require co-ordination between moving robots to deploy an effective on-field strategy. However, this is a design challenge for PiccSIM developers. Unlike most applications where the nodes are static and the position is set before the simulation (e.g. building automation or wireless process control), mobile scenarios requires dynamic position updates. Since PiccSIM uses two computers: one for network simulation (Ns2) and another for the dynamic simulation (xPC Target), the mobile wireless node positions has to be synchronized between these machines.

The positions of the mobile nodes are simulated in the xPC Target, depending on the dynamics of the robots and the position control algorithm. The position control signals could e.g. come from a remote controller over the wireless network. Thus the actual position of the robot would depend on the performance of the simulated wireless control system and be determined at runtime. This necessitates the need for a dynamic update of the mobile node positions during runtime. This enables the simulation of any wireless mobile scenario with moving robots or vehicles. The communication and control can co-simulated with the mobility of the wireless nodes taken into account in the simulation of the network.

In this paper we extend the scope of the PiccSIM simulator to enable simulation of mobile applications. We introduce a dynamic position-control model in Ns2. The integration of this model in PiccSIM allows users to remotely control node movement in Ns2. Our approach is to inform Ns2 of the position update by sending a UDP packet to the Ns2 computer to a specific port (22300) with information of the new node position. The format of the packet is shown in Fig. 4. The data contains the ID of the node to update and the new position (X and Y coordinates). Ns2 using its live network function taps the packet from the network and binds the position information to the intended node. The actual position update in Ns2 is done once every second.

III. CONTROL WITH BREAKING COMMUNICATION

In a mobile wireless scenario there is the possibility of link breaks when the mobile nodes move out of radio range. The communication will exhibit sporadic breaks when rerouting is performed. This is a new scenario not present in traditional control applications with fixed fieldbusses. If the control loop is broken the actions to be taken of the control system needs to be considered.

From the control point of view, long communication breaks are not desirable. A control loop can cope with heavy packet loss and there are theorems that still guarantee stability of the control loop [4]. But if consecutive packet losses are long, control becomes difficult. This is further elaborated here and some things which must be implemented in a control

HEADER	NODE ID	Position X	Position Y	PADDING

Figure 4. Format of node position update packet

system with breaking communication are presented.

Consider a regular control loop distributed over a (wireless) network which may have long communication breaks. During short breaks or packet losses (individual packets are lost or breaks in the order of the sampling time of the control system), little impact is seen on the control system, if it is designed to cope with these.

There are two approaches to cope with unreliable communication in control systems. One is to use a regular controller, e.g. a PID [11] or a state-feedback controller [5] and tune it to be robust to missing measurement packet. The other alternative is to use an estimator, e.g. a Kalman filter [13], [6], to estimate the current process state from intermittent measurements.

A. PID Controller Tuning for Delay Jitter

A controller can be tuned to tolerate additional delay caused by dropped measurement packets in the networked control loop. The jitter margin defines the amount of additional delay that a control system can tolerate without being unstable. The delay $\delta(t)$ can be varying in anyway, provided that it is bounded by δ_{max}

$$0 \le \delta(t) \le \delta_{max}. \tag{1}$$

The theorem [4] says that in the continuous case the plant $P(s)$ and controller $C(s)$ is stable if

$$\left|\frac{P(j\omega)C(j\omega)}{1+P(j\omega)C(j\omega)}\right| < \frac{1}{\delta_{max}\omega}, \quad \forall \omega \in [0,\infty]. \tag{2}$$

A PID controller tuning method for varying delay control systems using the jitter margin is the method by Eriksson [3]. The basics are briefly repeated here. Given a first order lag plus integral plus delay approximation of a process (a so called FOLIPD model)

$$G(s) = \frac{k_v}{s(T_f s+1)}e^{-sL}, \tag{3}$$

where L is the process delay T_f the time-constant and k_v is the process gain, of the process approximation. The desired jitter margin δ_{max} determines the PID controller tuning for integrated processes through [3]

$$\begin{cases} k = \dfrac{a}{k_v L} \\ k_i = 0 \\ k_d = \dfrac{aT_f}{k_v L} \end{cases}. \tag{4}$$

$$a = \frac{0.947362L}{\delta_{max} + 0.634629L} \tag{5}$$

The control system with a controller tuned this way can tolerate any excess delay that is smaller than δ_{max}, without risk of instability. This protects the control system against individual packet drops. The parameter *a* gives the maximum gain of the tuning for the given jitter margin. There are other similar tuning methods, each giving the gain *a* through different formula depending on the design goal.

With the maximum PID tuning gain obtained from (5), the response of the control loop might oscillate and be undesirable. We propose to include a tightness factor α to the tuning. By introducing a tightness factor α that modifies the gain such that the new a' gain is

$$a' = \alpha\frac{0.947362L}{\delta_{max} + 0.634629L}, \quad \alpha \in [0,1] \tag{6}$$

the tuning is more robust and the response is smoother.

B. Long Communication Breaks

The previous tuning gives stability in case of short breaks of single packet losses. Long breaks on the other hand are detrimental. If the controller does not receive any measurements of the process or the actuator does receive any control messages for a long time, the control system cannot take any action. The issue is then what to do. The process must be kept at a safe state: a stop action must be devised.

Usually the previous input is held until the next sample is received. This increases the delay temporary by one sampling time if a packet is lost. The varying delay is called delay jitter. If the process contains an integrator, the output will increase to infinity if a nonzero input is held. Therefore the input to the process should be set to zero if no control is received. If the process has no integrator, the input should be held and the process (if stable) will eventually reach this value. If the process does not include an integrator, the controller is usually designed with integration action to remove possible steady-state bias. In this case the error signal to the controller must be set to zero if no measurements are received. Otherwise there is an integrator windup.

The effects of breaks in the control loop are best demonstrated with pure simulations. A process model of $G_1(s) = \frac{2}{s+1}$ and for integrating processes $G_2(s) = \frac{2}{s}$ is selected. A PID controller of the form $C(s) = P + \frac{I}{s} + Ds$ is in the feedback loop with $P = 1$, $I = 1$ and $D = 0$ for G_1 and $P = 1$, $I = 0$ and $D = 0.1$ for the integrating process, G_2. The process and controller tuning are selected for demonstration purposes only.

The cases with communication breaks in the control loop are simulated with both keeping the last value as well as setting the input value to zero at time-instant $t = 1$, when the control loop is opened. The communication break can occur at both the path between the sensor and controller (s-c) and between the controller and the actuator (c-a). Both cases are considered and displayed in Fig. 5.

The only time when the previous value should be kept is when the process does not have an integrator and there is a communication break between the controller and the actuator (upper left case in Fig. 5). But in this case there will be an integrator windup in the controller (not evident from the figure). Still some anti-windup scheme must be used.

There must be a threshold, d_{max}, when to switch from regular control to the stop action. If one uses the delay jitter tuned controller the natural limit is when the delay increases beyond the maximum jitter, $d_{max} = \delta_{max}$. A suitable value for δ_{max} is such that the most of the packet drops causes less delay than δ_{max}. In the rare cases when the delay is larger, e.g. when rerouting is done at the network layer, the control system switches to the stop mode.

C. Kalman Filter with Missing Measurements

Instead of using a specially tuned PID controller as in the previous subsection a Kalman filter (KF) can be used to predict the missing measurements. This enables a regularly tuned PID controller to be used. The KF assumes a state-space system model of the form

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{w}(k) \\ \mathbf{z}(k) = \mathbf{H}\mathbf{x}(k) + \mathbf{v}(k) \end{cases} \tag{7}$$

where $\mathbf{x}$ is the state, $\mathbf{u}$ the control input and $\mathbf{z}$ the measurement vector. The noise terms $\mathbf{w}(k)$ and $\mathbf{v}(k)$ are independent zero

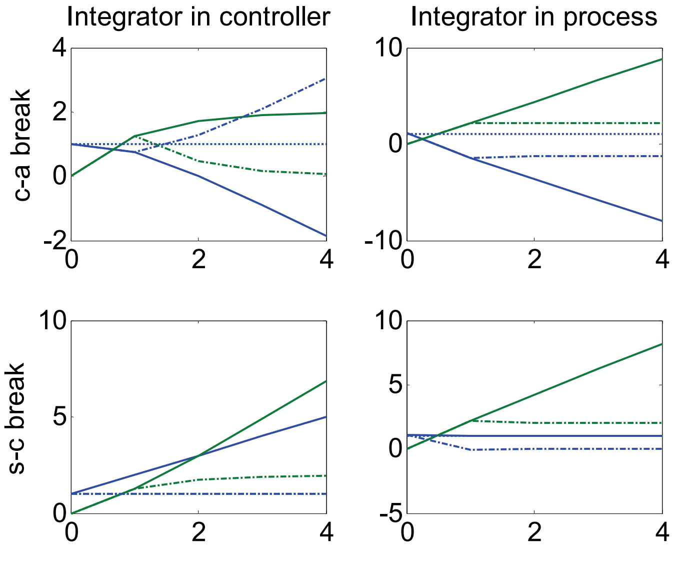

Figure 5. Control with breaks in controller to actuator or sensor to controller paths. Both for integrating processes and integrator in controller. Results with hold previous value (solid line) or set to zero (dashed line) for all scenarios. Output: green and control: blue.

mean Gaussian white noises with covariances $\mathbf{W}$ and $\mathbf{V}$, respectively. The KF is calculated with the standard prediction and update equations found in the literature, such as [7].

If the varying network delay is smaller compared to the sampling time, meaning that most of the packets are well received before the next sampling time, there are no problems of varying delay, only missing measurements need to be taken care of. It is important, that the Kalman filter can handle the dropped packets, since these are presumable in wireless systems. The case when there is a dropped packet, is equivalent to a measurement with an infinite variance [6]. As the corresponding diagonal element in $\mathbf{V}$ tends to infinity, the matching column in the Kalman gain $\mathbf{K}$ tends to zero, meaning that no correction is done in the filter with the missing measurement. Only the update part of the Kalman filter algorithm is done, and the state is estimated, for those cases when no measurement is available.

If a Kalman filter is employed to estimate missing data, no stop mode is needed for the control system during regular use, since an estimate is always available, even if the network has a long outage. If the network outage is really long (a total collapse of the network), the control system must eventually be shut down. This limit depends on the accuracy of the estimation.

D. Routing Protocols

In wireless multi-hop network, routing is considered one of the major issues. Routing protocols have to take into account factors as limited bandwidth, variable capacity of radio links, and energy-constrained operation of the nodes. A comprehensive survey of routing protocols developed for numerous purposes is in [1]. There are different approaches depending on the design choices and application. In our case the design target is a network, robust to node mobility and hop independent performance. It is not a trivial task to find a path from one node to a possibly distant destination node, considering the heterogeneous and dynamic nature of the environment and the possibility to only communicate with its neighbours. Also mobility, traffic requirements and harsh channel conditions accounts to link breaks requiring additional re-routing.

The frequency and length of re-establishing paths are important measures for the control system performance. If the packet drop length (delay) exceeds the threshold of the control loop, it switches to the stop mode. The frequency and length of these rerouting outages are important measures for the control system of e.g. wireless mobile robots. We want to minimize both the number of path losses and the length of recovery.

In this paper we have selected Adhoc On-demand Routing protocol (AODV) and Localized Multiple Next-hop Routing (LMNR) as the test protocol for single and multipath routing respectively. In static or low mobility scenario, it is obvious that having multipath routing provides resilience against link failure by fast recovery with backup routes [8]. But in case of high mobility applications and short radio (ZigBee), the advantage of having multiple paths could be negated by the high frequency of link failures mostly due to mobility. It would then be beneficial to use single path or periodic routing, over multipath routing. In this paper we examine the behavior of multipath routing protocol with ZigBee radio in a high mobile radio application i.e. wireless robot squad.

E. Packet Priorization Based on Hop Count

Prioritization has always been a genuine requirement in many sensor network applications. Information can be categorised as critical (time constrained) and non critical data. The scenario becomes worse when the bit rate for non-critical data is high and the routing protocol does not support data prioritization. Measurements can be considered less vital and even if some packets are dropped. Robust control algorithms (jitter margin tuned PID controller) or the present state can be estimated (using the Kalman filter). However the overall control performance should be good. Every control loop should get equal quality of service of the network, even if some loops stretches over longer paths. In this paper we examine data prioritization in context of controller performance.

We use and evaluate two *hop* based prioritization methods. We propose a forwarded hop based priorization strategy. The idea is to compensate for the longer end to end delay by increasing priority of those packets, which have been forwarded the most. The heuristic of prioritizing based on travelled distance is that the network has invested resources on getting it this far, so it has an incentive to get the packet all the way to the destination, or the effort is wasted. The second strategy is to prioritize based on the number of total hops for a given source-destination pair. Packets destined to travel far are prioritized.
The implementation for data prioritization is done in Ns2, here instead of using First in First out queue; a simple sorted queue algorithm is used.

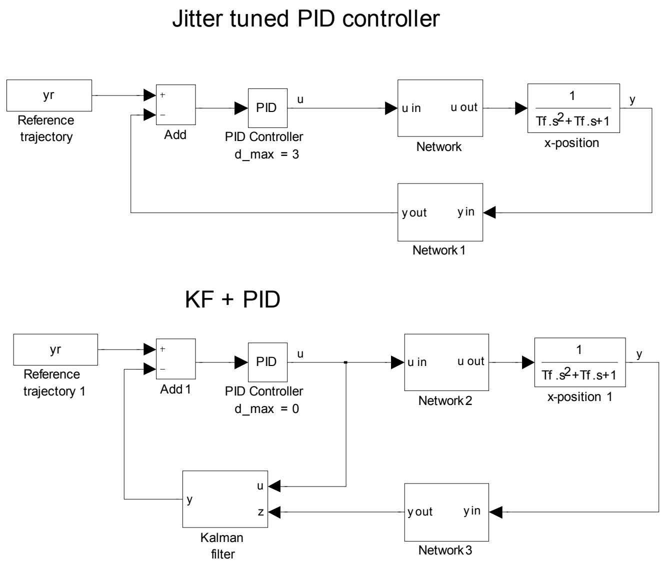

Figure 6. Control loop for jitter tuned PID controller and for Kalman filter and PID controller.

IV. SIMULATIONS

A. Scenario

The simulated scenario considers a squad of mobile wireless robots. They move in different formations and change the formation from time to time. There is one leader robot which controls the positions of the other robots. It is assumed that the robots can locate themselves e.g. based on GPS, odometer or inertia measurements. The robots send their position to the leader robot. The leader then calculates the desired path and control signals, taking into account collisions and the final formation. It then sends the control message to the other moving robots at every sampling time.

The network performance and ultimately the control performance depend on the formation of the robots and how the packets are routed through the network. As the robots change formation the links may break and a new route must be established. The speed of which the path is re-established depends on the routing protocol. The communication outages degrade also the control performance and the issues previously discussed are implemented and tested.

The communication conditions are modeled with Ricean fading with parameter $K = 20$. This means that there will be individual packet losses because of lossy links, on top of the link break due to mobility. The interval of Hello messages is h. After three hello message drops a link break is detected.

The dynamic model of the location of the robots is the following in both the x- and y-direction

$$G(s) = \frac{U(s)}{Y(s)} = \frac{1}{s\left(T_f s + 1\right)} \tag{8}$$

where y is the position and u the control received from the controller (zero order hold is assumed between the reception of control packets) and $T_f = 25$ s is a time-constant. The velocity of the robot is constrained to $\left|\dot{y}(t)\right| \leq 0.6$ m/s. If a robot does not receive a control message for 3 ($d_{max} = 3h$) sampling times, the stop action is switched on and a local controller halts the robot (controls the velocity to zero), until new control messages are received.

The position controller for each mobile robot is a discrete-time PID controller with sampling time h, tuned with the jitter margin method. A delay of less than one sampling time is assumed: $L = h$. The switch-point between control and the stop mode is selected to be when no new packet has been received for three consecutive sampling times, $d_{max} = 3h$. This threshold is selected because if three consecutive packets are lost, a reroute request is issued by the routing protocol, and a longer break is expected. The control system is thus coupled with the communication. The other controller parameters are $\delta_{max} = d_{max}$ and tightness factor of $\alpha = 0.5$. This will give satisfactory control performance even if the delay is at the maximum value.

The control using a Kalman filter to estimate the current position and a PID controller is also simulated. The robot dynamic model (8) is discretized for the Kalman filter and brought to state-space form. The covariances are selected as diagonal matrices with 0.1 on the diagonal for **W** and 0.01 for **V**, respectively. Since the Kalman filter estimates the current process state, there is no delay jitter seen at the PID controller. For comparison a similar tuning as with the previous case is selected. The controller is tuned with the same parameters, but zero jitter margin $\delta_{max} = 0$ is used, which will give a tighter and better tuning. Both control loops are depicted in Fig 6.

The control performance is calculated with an integral error cost function. Since a link break followed by stop action results in a large deviation of the desired trajectory, this will dominate the cost function. Therefore the cost is calculated as an average of only the instances when the communication link is working according to

$$J = \frac{1}{K}\sum_{k=1}^{K}\frac{\sum_{i=0}^{T/h-1} h\left\|\mathbf{r}_k(i)-\mathbf{y}_k(i)\right\| I_k(i)}{\sum_{i=0}^{T/h} I_k(i)} \tag{9}$$

where $\mathbf{r}_k$ is the reference trajectory, $\mathbf{y}_k$ the position of the k^{th} robot and I_k is an indicator function indicating if there is a break in the communication

$$I(i) = \begin{cases} 1, & \text{Communication} \\ 0, & \text{Break in communication} \end{cases}. \tag{10}$$

A break in the communication is defined as when three consecutive packets are dropped, until the reception of a packet.

Simulations of three formation changes of a squad of $K = 25$ robots are done. The formations are shown in Fig. 7. The robot reference trajectory between the two formations is a cubic spline with an endpoint constraint of zero velocity. A T = 1060 seconds simulation with every routing protocol and priorization is done. The three-formation change pattern is repeated two times, leaving 150 seconds for every formation change, plus some extra time for the initial organization and settling down after the final formation change.

Next the results for singlepath/multipath routing protocol and some priorization alternatives are compared. The results are given for both the jitter margin tuned PID and the KF+PID combination.

B. *Single path vs multipath routing protocol*

The robot formation change scenario presents a unique test case for the adaptability and the impact of using multipath routing in mobile applications. However the simulation results suggest that the high frequency of link failure due to continuous mobility of the robots and channel model characteristics has mitigated the advantages of multiple paths in LMNR and similarly increases the control cost. Table 1 reflects an increased in packet loss (%) for LMNR over AODV. In this case using single path routing would be a more preferred choice. However there is an increase in routing overhead.

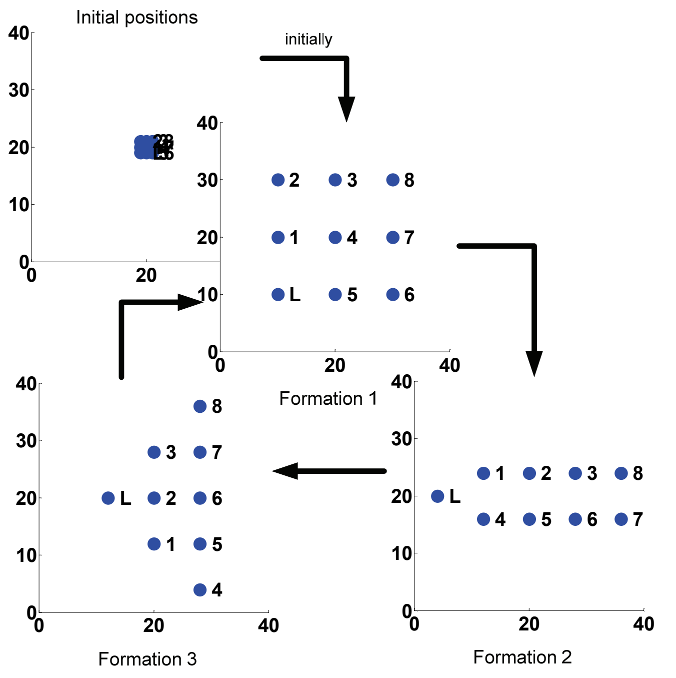

Figure 7. Formations in simulation. The leader node indicated by "L". The three formations are looped 10 times.

AODV makes more route request than LMNR, thus having fresher routes and this in turn contributes to decreased delay. To conclude, using single or multi path routing depends on application requirements. There is a trade-off between energy efficiency and control performance. AODV provides better packet delivery and eases controller design with lower average delay on the expense of additional routing overhead.

The control cost is tabulated in Table II. Using singlepath routing is also more favorable as it results in slightly smaller control cost, regardless of the control algorithm.

C. *Packet prioritization*

Packet prioritization relaxes controller design requirements by equalizing packet delays for different number of hops. In this paper we have presented two prioritization metrics: forward count and total hops. The former uses the number of times a packet has been forwarded and the latter number of total hops for a given source-destination pairs. The basic idea is to compensate for packet delay for different hop counts by prioritizing packets using one of the above metrics in a sorted queue algorithm.

Fig. 8 and 10 shows the average end to end delay for singlepath, respectively multipath routing using the forwarded count priorization. Using priorization equalizes the delay for multiple hops with both control strategies. This means that the

TABLE I. PERFORMANCE METRICS FROM SIMULATIONS

	Avg. delay [s]	**Routing overhead [%]**	**Packet loss [%]**	**Avg. length of outage [packets]**
AODV	0.06	4.05	43.18	7.7
LMNR	0.197	2.99	48.83	5.1

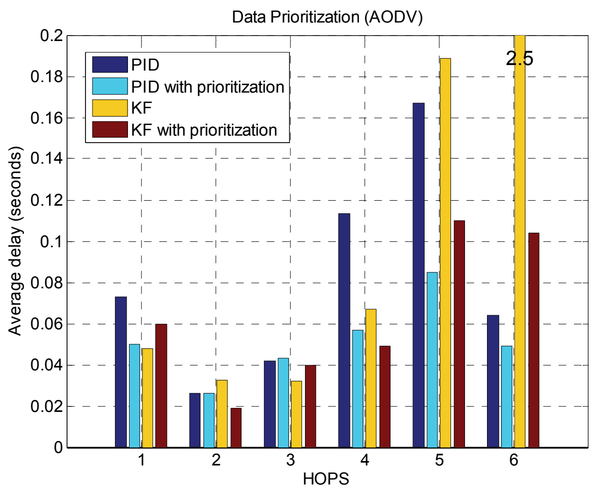

Figure 8. Packet delay for singlepath routing as a function of hops.

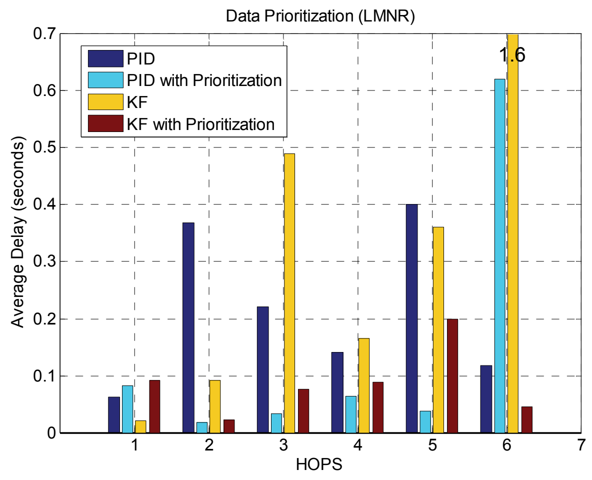

Figure 10. Packet delay for multipath routing as a function of hops.

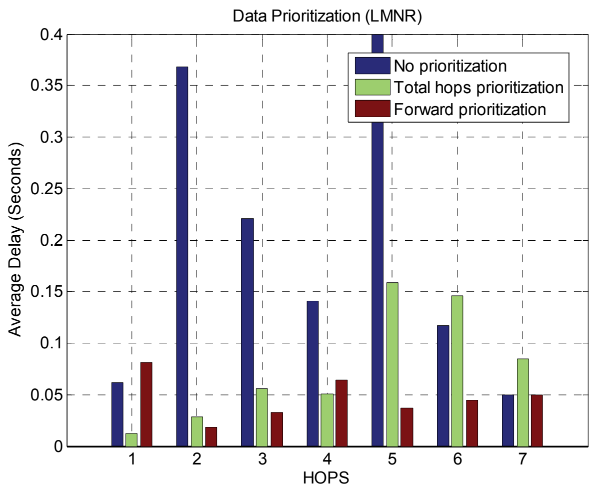

Figure 9. Average delay as function of number of total hops, with different priorizations.

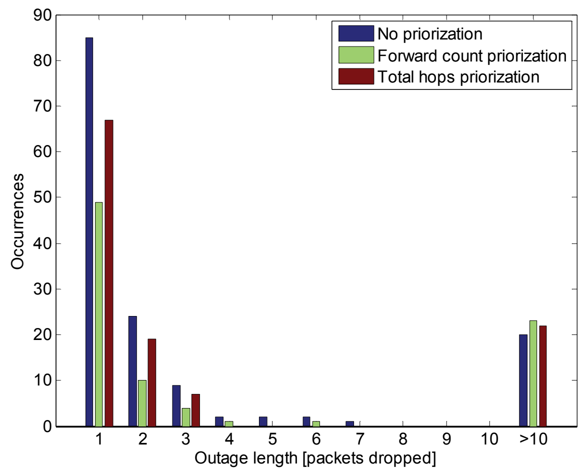

Figure 11. Histogram of outage lengths (packets dropped) for different priorizations (with LMNR).

far away node is not penalized, by a larger delay, for being far away.

In Fig. 9 the average delay is compared between no priorization and the two priorization strategies as a function of hop count. Priorization based on forwarded count results in equalized average delay regardless of the number of hops. This is an advantage for the control system, since in this case it does not have to take the number of hops (which might change dynamically) into account. Priorization based on total hops delivers more packets to the far away nodes (not shown) and thus the average delay is larger. This is also favorable for the control system. The control designer should decide if decreased delay variation or better packet delivery is more desired.

In order to make a comparison between the two prioritization methods from the breaking of communication point of view, we have introduced a new metric: outage length, which indicate the number of packets dropped when a connection between the robots is lost. Fig. 11 presents the outage length for all priorization methods. The difference in packet drops can be explained as the total hop metric prioritizes packets with higher hop where as forward count increases the priority as the packet transverse through the network. This however increases the routing request made by total hop priorization, as the probability of link failure for farthest robots is higher. This will contribute to higher packet delivery for farthest robots, thus increased fairness and on the flip side, more routing overhead. In the figure outage length > 10 denote that robot is out of formation with no neighbors.

The control costs (9) are tabulate in Table II. The jitter tuned PID controller is better than the Kalman filter, although

TABLE II. CONTROL COST WITH BOTH CONTROL ALGORITHMS, FOR SINGLEPATH AND MULTIPATH ROUTING PROTOCOL, WITH AND WITHOUT PRIORIZATION.

Protocol	Jitter PID	Kalman filter + PID
AODV	18.1	21.1
AODV + Priorization	17.7	14.0
LMNR	18.9	21.8
LMNR + Priorization	18.1	19.2

the latter enables a better tuning and faster control. The explanation is that once a robot gains communication, it catches up the reference so fast that it soon moves out of range again. The jitter PID controller has looser tuning and the robots move slower, enabling the maintenance of the route to the mobile robot. Using the singlepath routing protocol results in smaller control cost, as the network quality of service is better. The increase in cost of multipath routing is recovered by including priorization.

V. Conclusions

We presented a simulation system, PiccSIM, for evaluating the control and communication performance of wireless control systems. We have developed an extension to PiccSIM which enables the simulation of mobile wireless robots. The position of the wireless nodes can be dynamically changed during the simulation. The location of the robots and link breaks caused by mobility determines the network performance, as the topology of the network changes. The communication and control system exhibits breaks because of the topology changes. The tuning of the control system with an unreliable network and the necessary tasks to do when the control loop is broken were discussed and demonstrated. The PiccSIM simulation platform makes such joint control and communication studies easy to perform.

The example case of a mobile robot squad was simulated to highlight the issues with control over breaking communications. The priorization of packets was considered, to give an equal quality of service to the control system, regardless of the hop distance to the destination. Using a singlepath routing protocol is advantageous over a multipath protocol because of the larger number of link breaks of the latter. This conclusion is valid only for routing over a network with frequent topology changes. A general result is that priorization based on forward count equalizes the delay to the far away nodes and improves the overall control performance. The control performance results and conclusions are general and apply to all control systems with breaking communications due to mobility.

References

[1] K. Akkaya, M. Younis, “A survey on routing protocols for wireless sensor networks”, Ad Hoc Networks, vol 3, iss. 3, Elsevier, 2003.

[2] Al-Hammouri, D. Agrawal, V. Liberatore, H. Al-Omari, Z. Al-Qudah, and M. S. Branicky, “Demo abstract: a co-simulation platform for actuator networks,” Sensys 2007.

[3] L. Eriksson and T. Oksanen, “PID controller tuning for integrating processes: analysis and new design approach,” Proc. of the 4th International Symposium on Mechatronics and its Applications (ISMA07), Sharjah, U.A.E. March 26-29, 2007.

[4] C.-Y. Kao and B. Lincoln, “Simple stability criteria for systems with timevarying delays,” Automatica, vol. 40, pp. 1429 – 1434, 2004.

[5] H. Li, Z. Sun, M-Y. Chow, and B. Chen, “State feedback controller design of networked control systems with time delay and packet dropout”, Proc.17th IFAC World Congress, Seoul, Korea, 6-11 July, in press.

[6] X. Liu and A. Goldsmith, “Kalman filtering with partial observation losses,” 43rd IEEE Conference on Decision and Control, vol. 4, 14-17 Dec. 2004, pp. 4180-4186.

[7] P. S. Maybeck, Stochastic Models, Estimation and Control, vol. 1, Academic Press, 1979, pp. 229–230.

[8] S. Nethi, M. Pohjola, L. Eriksson and R. Jäntti, “Simulation case studies of wireless networked control systems,” Proc. 10th ACM/IEEE International Symposium on Modelling, Analysis and Simulation of Wireless and Mobile Systems (ACM/IEEE MSWiM 2007), Crete, Greece, Oct 22-26, 2007.

[9] S. Nethi, M. Pohjola, L. Eriksson and R. Jäntti, ”Platform for emulating networked control systems in laboratory environments,” The 8th International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2007, Helsinki, Finland, June 18-21, 2007.

[10] Ns-2. The Network Simulator, http://www.isi.edu/nsnam/ns/, Referenced 22th of January 2008.

[11] M. Pohjola, L. Eriksson and H. Koivo, "Tuning of PID Controller in Networked Control," Proc. 32nd IEEE Industrial Electronics Conference, Paris, France, 7-10 November, 2006.

[12] M. Pohjola, L. Eriksson, V. Hölttä and T. Oksanen ”Platform for monitoring and controlling educational laboratory processes over internet,” Proc. 16th IFAC World Congress, Prague, Czech Republic, 4-8. July, 2005.

[13] L. Schenato, "Optimal estimation in networked control systems subject to random delay and packet loss," 45th IEEE Conference on Decision and Control, 13-15 Dec. 2006.

[14] Y. Wei, R.T. Vaughan, G.S. Sukhatme, J. Heidemann, D. Estrin, M.J. Mataric, "Evaluating control strategies for wireless-networked robots using an integrated robot and network simulation," Proc. IEEE International Conference on Robotics and Automation, vol.3, 2001, pp. 2941-2947.