UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

SCHEDULED MESSAGE DELIVERY IN DELAY-TOLERANT NETWORKS

Csaba Kiss Kalló and Mauro Brunato

February 2004

Technical Report # DIT-04-014

Scheduled Message Delivery in Delay-Tolerant Networks

Csaba Kiss Kalló
Mauro Brunato
Università di Trento
Dipartimento di Informatica e Telecomunicazioni
Via Sommarive 14, Trento, Italy
[kkcsaba,brunato]@dit.unitn.it

Abstract

With the proliferation of mobile communications, new kinds of network architectures are being defined over the existing static one. Delay-Tolerant Networks (DTN) come to fill in a gap in the existing networking technology: they provide support for communication between peers when there is no end-to-end path available between them.

Message forwarding in DTNs can be handled in many ways. In this work the performance of a DTN is analyzed in the case of scheduled message delivery, with particular consideration to the different message delivery delays and to the usage of buffers that store a message during its propagation from its source to the destination.

We introduce the `K2` heuristic algorithm for location-aware message delivery, based on the k-nearest-neighbors technique, and compare simulation results with another basic message delivery technique, Message Forwarding with Flooding (`MFF`).

The results of simulations on a simplified urban setting show that `K2` reduces buffer usage while maintaining delivery delays approximately unchanged when compared to `MFF`. Simulations also suggest that adding a DTN-like message forwarding architecture with mobile access points to an existing fixed infrastructure can be an effective way to improve wireless coverage for delay-tolerant end-user applications such as e-mail, message passing and news broadcast.

Keywords: delay-tolerant networks, scheduled message delivery, simulation, heuristic algorithms

1 Introduction

Pervasive networking environments, with always-on wireless connectivity by means of portable computers, PDAs and cellular phones, are becoming part of our everyday experience, at least in office environments. A trend towards the everyday life has however begun. Low-bandwidth systems such as packet-oriented cellular connections (GPRS) have been commercially available for years, and third generation cellular systems, oriented both to voice and data, are being rapidly deployed and commercialized.

On the other hand, wireless LANs are starting to appear in many public places, such as stations and shopping malls. As extensions of traditional local-area networks, WLANs are used to carry TCP/IP-like traffic, with the underlying hypothesis that an end-to-end connection between peering devices is available.

A different paradigm, based on a family of store-and-forward techniques, is at the basis of Delay-Tolerant Networks (DTN). In this paper we propose and analyze through simulations some techniques devised to provide delay-tolerant services, such as mail and messaging, to users that are not directly connected to a wireless LAN, by forwarding information by means of mobile access points.

Simulations take into account a user-related parameter, namely the message latency, and an important network-related parameter, the maximum size of the buffers in the mobile and fixed nodes of the network.

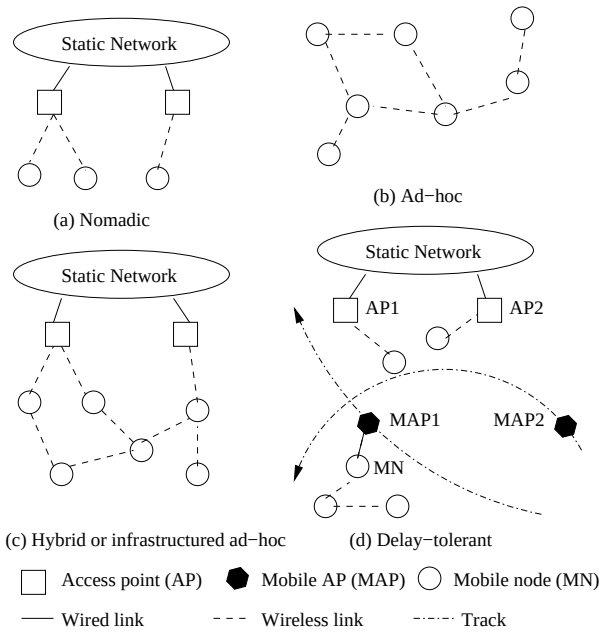


Figure 1: Wireless network architectures

This paper is structured as follows. In Section 2 the concept of DTN is introduced by examining relevant literature. Section 3 defi nes the exact terms of the problem. In Section 4 our approach to the problem is described, together with the proposed heuristic algorithms for scheduled delivery. Section 5 describes the implementation of our simulator while in Section 6 some results of our simulations are reported and discussed.

2 Related Work

With the proliferation of mobile communications, new kinds of network architectures are being defi ned over the existing static one [1]. The simplest architecture is called *nomadic* and consists of a traditional static network having access points at its periphery that mobile wireless devices use to access the network, as shown in Figure 1.a. On the other hand, the so-called *ad hoc* networks are exclusively formed by mobile components that connect to each other through wireless links, as in Figure 1.b.

In a latter moment, from the combination of these two basic kind of mobile wireless netwok architectures a *hybrid* or *infrastructured ad hoc* network has been defi ned [2, 3]. The difference between nomadic and hybrid networks is that while in the former case only one-hop connections to the access points are allowed, in the latter architecture multihop links toward the access points are also permitted (see Figure 1.c).

Due to mobility and to the reduced range of wireless networking technologies, some devices or groups of devices may move away from the main networking area, thus *partitioning* the network in clusters. Devices in a cluster form an isolated network segment, without access to the resources available in the other parts of the network and to potential communication peers. To make inter-cluster communication possible, two main streams of approaches are available in the literature. First, communication links between two clusters, or a cluster and the Internet, can be set up using specialized gateways with enhanced radio capabilities and/or airborne relay nodes (e.g. airplanes, satellites), like in the systems targeted by [4, 5]. In this architecture devices are partitioned in two or more layers that take advantage of

heterogeneous routing algorithms for managing communication. Thus, these networks form a hierarchical topology with *ad hoc clusters* (i.e. partitions of ad hoc networks) at the lowest layer. Superior layers may also have the role of *translators*, enabling different communication technologies to be present in the same network. Albeit this solution is simple from a technical point of view, it is very expensive.

In the second approach, mobile nodes that are expected or scheduled to move among clusters contain a bundle with the inter-cluster communication data. Routing in these networks is done on a *store-and-forward* basis, i.e. the data is stored at every node on the route, until another node closer to the destination can overtake it. In these networks an end-to-end path between source and destination is not required. This kind of communication paradigm implies both data and device traffic. A communication system following such a model is called a *Delay-Tolerant Network* (DTN) [6, 7]. Delay-tolerant network functionality is implemented in the so-called *Bundle Layer*, inserted on top of the transport layer in the traditional networking protocol stacks. Such systems are addressed in [8, 9]. Projects like the InterPlaNetary Internet [10], the Saami Network Connectivity [11] and others build their specific applications on DTNs. An example for DTN architecture is presented in Figure 1.d, which shall be fully explained in Section 3.

Delay tolerant networks were introduced to solve several problems of traditional networking technologies. One of the main issues to be mentioned here is that current technologies offer poor support, if any at all, for communication between two network elements if no end-to-end path is available. Long propagation delays and high error rates make proper functioning of the most common networking protocols (TCP, SCTP, UDP, IP, SNMP, RIP, BGP, ...) impossible, since frequent retransmissions, expiration of the 225s long TTL, links marked as non-operable, and so on, make them abort all initiated communication channels [7]. The store-and-forward model of DTNs is able to overcome all these problems.

3 Problem Formulation

In our work we investigate the performance of a DTN in an environment with many message forwarder nodes that move periodically. We are mainly interested in the delays between the moment of sending and receiving a message and the size of the different buffers on the route required for temporarily storing the message. On this purpose:

- we describe a possible application environment
- we define K2, a novel message handing mechanism
- we implement a simulator for the application that enables us to perform experiments with our message handing mechanism

The central element of our application is the public transportation network of a city. The vehicles of such a network, equipped with a gateway device, collect data bundles from mobile users they meet on their way on the streets of the city and forward this data to the static data communication network when arriving in the range of a fixed gateway. On the reverse direction, vehicles could transport data also from the infrastructure to mobile users, in the same manner.

Figure 1.d illustrates how a mobile node MN (e.g a wireless PDA owner walking on the street), within an ad hoc cluster, places a query on a mobile access point MAP1 (installed, for instance, on a bus). After a while, MAP1 moves close to the static access point AP1 (set up in a bus stop), and forwards the message of MN toward the destination

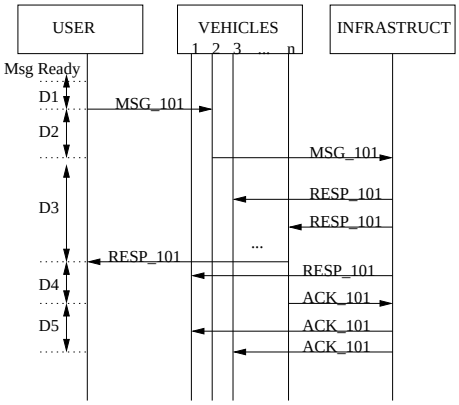


Figure 2: Message flow between users and the infrastructure

of the data, e.g. an e-mail server. After the reception of the message the e-mail server elaborates a response and multicasts it to AP1 and AP2. A short time after, MAP2 (another gateway-equipped bus), passes close to AP2 and takes the message back to the area where MN placed its query on MNP1. When arriving in communication range with MN, MNP2 hands the respose data, i.e. several email messages.

The installation of gateway devices on the vehicles of such a transportation network can provide network connectivity also in places where the deployment of static access points is unprofi table. Indeed, the advantage of this network architecture is its capability of expanding the coverage of a wireless network on a considerably bigger area than with the same number of static gateways, at the same cost. The price payed for this is the intermittent connectivity of such a network that implies longer response times that the mobile user will experiment in user-initiated transactions. The length of these response delays is one of the issues that we investigate in this work. Since these delays lead to the accumulation of data on both mobile and static gateways, we also want to obtain from our experiments a clear vision upon the appropriate size of the buffers on these devices that can assure the proper functioning of the system.

The objective of this work is to optimize scheduled message delivery in DTNs without signifi cant negative influence on the message delivery delays that users experiment. This issue is discussed in details in the subsequent sections.

4 Approach

We focus our attention on the network segment dedicated to provide the connection between a mobile user and the infrastructure. In other words, in this phase of our work we do not consider delays or networking issues such as routing behind the static access points, i.e. in the static network, with the assumption that these times are usually much shorter than the physical delivery time by vehicles. Our main concern is to move data between the mobile users and the access points[1].

The network performance is analyzed with respect to the mentioned parameters (i.e. forwarding delay and buffer length) in each of the two message forwarding algorithms, MFF and K2, described in the following subsections. Both algorithms follow the generic communication scheme presented in Figure 2, as explained next.

[1]Infrastructure, static network and the access points in the generic sense are used as synonyms in this work, if not specifi ed otherwise.

Let us suppose that a mobile user wants to send a request to a server in the static network. After the request has been issued on the user's device, at the first opportunity, i.e. when a bus arrives in his communication range, the message is sent to the bus with a unique identifier (`MSG_101`). The bus holds the message until a static access point appears in range. At this moment `MSG_101` is forwarded to the infrastructure and a response (`RESP_101`) is generated by the appropriate server. This response is replicated and forwarded to a set of buses that pass next to a static access point until a receipt acknowledgment (`ACK_101`) is received. A receipt acknowledgment is generated by the bus that actually hands `RESP_101` to the user, and it is sent to the infrastructure at the first opportunity. The infrastructure informs all the buses about the acknowledgment, so that they can remove `MSG_101` from their buffers. After the reception of the acknowledgment the response message is removed from the infrastructure. A Time-To-Live parameter can also be implemented to eliminate old messages from static and mobile access point buffers.

During this message exchange sequence we encounter at least five types of delays, indicated in Figure 2. The first delay (D1) takes place between the moment when the user application sends the request and the one when a bus arrives into the user's communication range to overtake it. During this period the request message is stored in the bundle layer [6] or a corresponding middleware buffer. During the second delay (D2) the message is in the vehicle buffer. This delay lasts until the bus meets the first static access point. At this moment the messages arrive into the static network. The round trip of the message between the static network boundary and the appropriate server, as well as the time necessary for the server to elaborate a response for the request introduce a new delay. Discussion on this delay is not in the scope of this work, and we shall assume that the response is nearly instantaneous when compared to the other delays.

The consequent delay (D3) lays between the moment when the response arrived back from the server to the access points and the one when the user receives the response. The delays after this moment are not sensed by the user anymore. They are important from the system's point of view, in particular when we want know how long the already delivered messages occupy the system resources. There are two delay periods that we mention after this moment. During the first one (D4) the infrastructure has not yet been informed that the response was delivered to the user. Therefore in this period the infrastructure keeps propagating RESP_101 among the buses, therefore occupying communication bandwidth and storage needlessly. Finally, in the last delay period (D5) the infrastructure propagates short acknowledgment messages enabling the buses to free their buffers. Numeric values of these delays can be found in Section 6.

Clearly, the mentioned delays have been introduced to take into account a complete query-response transaction originated by the user towards a server in the static network. If a message is originated within the static network and must be simply sent to the user (e.g., the user has a subscription to a news service) the total delivery time is just the D3 delay, since nothing has been generated by the user.

From the above presentation of the delay types, it turns out that the most critical segment of the data flow in our architecture is the one from the infrastructure to the mobile gateways. In fact, the number of message replicas grows significantly on this segment. Over the physical constraints that inevitably impose long delays, the message propagation policy on this segment also influences the round trip times of user messages. If the response message is replicated on many mobile gateways, it is expected that the user will receive it earlier. However, high number of replicas will have negative impact on our second parameter, the buffer occupancy, as well as on the communication

```
1.  forall messages m in AP_Queue_In do
2.      PushBufferOut(KNN(m))
3.      forward list ← KCC(m.id)
4.      remove m from AP_Queue_In
5.  forall messages m in each AP_Queue_Out do
6.      if destination in range of AP then
7.          hand m to destination
8.          ap.PushAckQueue(m.id)
9.          remove m from AP
10.     else
11.         for each bus b do
12.             if (b in range of knn AP) and (b ∈ forward list with m ) then
13.                 b.PushBufferOut(m)
```

Figure 3: Pseudocode of the K2 algorithm

bandwidth. Therefore, a trade-off should be found that assures acceptable delays with reduced number of replicas.

After the generic description of message forwarding in the specified kind of DTN, next in this section we present the particularities of two algorithms. In the first one the number of replicas is not limited by the system while in the second algorithm an optimized approach is presented.

4.1 Message Forwarding with Flooding (MFF)

Message forwarding with flooding (MFF) is a simple but resource consuming technique. The main characteristic of this technique is that the infrastructure does not make any attempt to reduce the number of replicas of RESP_101 that it propagates in the DTN, but it simply forwards these messages to each mobile gateway passing in its radio proximity. Every static access point will receive a replica of the message and, if no acknowledgment is received for a long enough period of time, all the mobile gateways will have the message in their buffers. Thus, replicas of the message will be stored also in the buffer of gateways that physically have no any possibility to deliver it to the destination. However, the big advantage of MFF is that if a single opportunity exists for the message to be delivered to the destination, this technique will not miss it.

4.2 The K2 Algorithm

In this subsection we define the K2 algorithm with the intention of improving the way resources are used when forwarding data in a DTN with scheduled mobile gateways. In K2 we use the *k-nearest neighbors* (KNN) technique in two different ways, hence the name of the algorithm.

The K2 algorithm is sketched in Figure 3. In order to reduce the number of replicas of the response message, the algorithm reduces the set of gateways that will receive the message by using location information. On this purpose the system will register and maintain the position information of the users. Notice that it is not required for the user device to have location estimation capabilities, but it is the (mobile) access point that takes care of location information. Naturally, if the user device has optional positioning capabilities, it may communicate that information to any gateway within range, which will propagate that data on to the infrastructure.

The K2 algorithm will use this information to ham in a region in which the user waits for the response with a high probability. In particular, it selects the K nearest static access points to the position found in the message, where K

is a parameter of the algorithm. The message will be copied only to the K selected access points (line 2), in contrast with MFF, where the message is replicated on each static gateway. In the fortunate case in which the user is connected to a static access point, the message is delivered immediately, bypassing the mobile gateways (lines 7–9). However, if the user is not within the infrastructure's reach, an additional algorithm is used for selecting the most appropriate mobile gateways that the message should be replicated to (line 3). On this purpose we apply the k-Nearest-Neighbors search for time instead of space and we call this variation *k-closest connections* (KCC). Indeed, with KCC we look for the identity of the first C buses that will have in their radio proximity the position where a message's destination was last seen by the infrastructure (note that the parameters of the two searches, K and C, may not be equal). A further requirement for a bus to be selected as a C-closest connection is that it must pass in the radio proximity of a K-nearest static access point (i.e. one of the K nearest access points selected for that message) before passing by the mentioned position. This constraint is necessary in order to assure that the bus will have the opportunity to overtake the message in question from the infrastructure.

5 Simulation

To test our algorithm and study buffer lengths and message delivery delays we implemented a simulator for a city bus scenario in C++. Next in this section this simulator is presented in details.

We start out by generating the elements of our scenario. In particular we generate the bus tracks on an area of $5000 \times 3000\text{m}^2$. Bus tracks consist of bus stop positions connected with straight lines. For each bus stop we assign a time between 10s and 60s that the buses following that track will spend there. Further, we set the average speed of $3 \div 10$m/s the buses will travel with on each segment between two bus stops. Finally, for each specific line we set the frequency the buses will follow each other. This frequency varies between 10 and 30 minutes.

User tracks are generated in a similar manner with some differences. A random starting point is generated for each user track, then new points are added close to the previous one that the user will reach with a walking speed of $1 \div 3$m/s. The angle formed by two consequent track segments is selected with care in order to simulate users that follow straighter paths or others that move back and forth on a small area. During the simulation users generate messages destined randomly to some other user or to the infrastructure that will answer them. Each user generates messages according to the Poisson distribution, with different average frequency.

Figure 4 shows the setup of some experiments. On the left, bus tracks are reported along with user tracks and static AP positions. On the right, a grayscale plot of coverage frequency is shown. White circles indicate full coverage, darker regions are covered for less than 100% of the total simulation time. Note that the scale is nonlinear in order to enhance the visibility of darker regions.

Our simulator starts out by loading the bus and user track data. From this data we first calculate the position of the fixed access points by placing one access point every third bus stop on each line, avoiding superposition in stops that are common to more lines. We also calculate the *round duration*, the time necessary for a bus to travel from one end of a line to the other and back. From the round duration value and the bus frequency we calculate the number of buses that will cycle on each line during the simulation.

Before users start to move on the map and send messages, buses are moved on their tracks for the duration of the longest round trip time, so that a steady mode of operation is reached, with as many buses per track as needed.

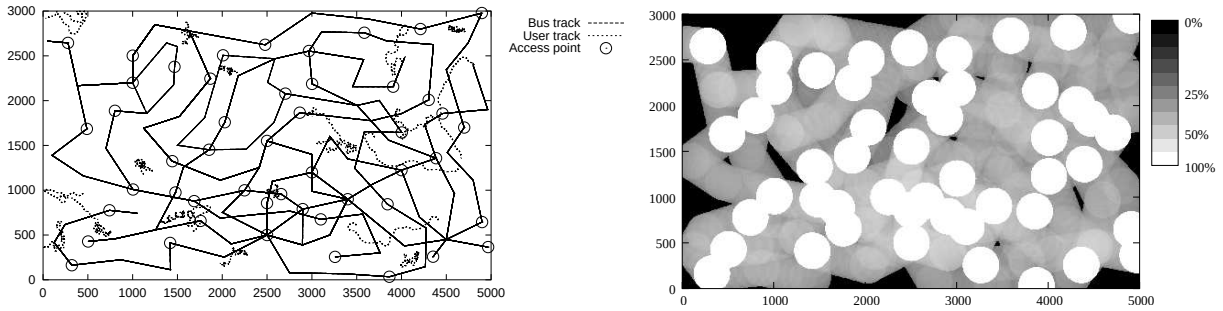


Figure 4: (Left) Bus lines and user tracks. (Right) corresponding time of coverage during the simulation. Note that the greyscale is not linear.

When the bus system is fully deployed, the main part of the simulation, handling users and message exchange, begins. The state of the system is evaluated every 0.5s of simulated time until the preset number of messages were received and removed from all buffers, or until a preset simulation time expires. Each step of the simulation starts by updating the locations of buses and users then user outgoing buffers are checked for newly generated messages. If in the buffer of a user u some messages are found, the system checks whether u is connected to any mobile or fixed access point, (M)AP for short. In case of available connection(s) all pending messages of u are forwarded to the appropriate (M)AP, giving higher priority to fixed access points.

MAP buffers are also checked at each iteration. If a bus arrives in the range of an AP it will forward all the messages it has in its incoming buffer (*BQin*) to the incoming buffer of the AP (*APQin*). In *BQin* there may be data or acknowledgment messages. The latter have no payload, therefore they only occupy a small space in the buffer, while messages containing user data are all of the same length. Acknowledgment message identifiers are stored at the APs, in order to keep track of acknowledged messages, and then deleted from *APQin*. After the forwarding each bus will delete from its *BQout* all the messages in the acknowledged message list of the infrastructure.

At this point messages are processed by the infrastructure and then forwarded or replied. Messages from the infrastructure to the users are sent using the `MFF` or the `K2` algorithm. Therefore, some messages will be handed directly to the destination by the APs while others will be replicated on a number of MAPs. Messages from the outgoing buffer of MAPs are handed to their destination when the respective user will be in range of the MAP. If this never happens, then that message will remain in the (M)AP buffers until the end of the simulation. In a future version of the simulator a time-to-live value may be attached to the packets enabling the automatic removal of old messages. Further, with the modification of the `K2` algorithm new attempts to deliver the message to its destination could be triggered by a timeout or some other event. These issues are subject for future work.

A message is deleted from the *BQout* of a MAP when it is handed to its destination. At the same time a new acknowledgment message is added to the *BQin* of the same MAP.

The infrastructure registers and maintains the location of each user that connects to the infrastructure. The `K2` algorithm uses this information to select the K nearest access points and the C closest connections of a particular user.

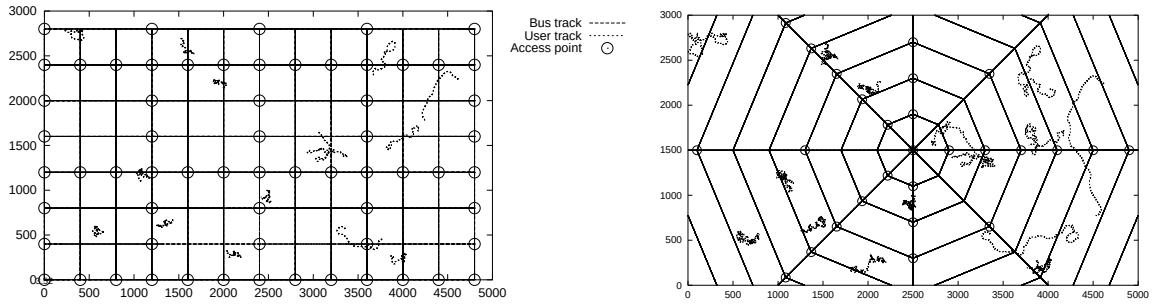


Figure 5: (Left) Grid map example. (Right) Radial map example

6 Simulation results

To evaluate K2 and get an overall view about the buffer sizes and message delivery delays we performed simulations with numerous different settings. The most important input parameters that we acted upon were the bus map (i.e. the set of bus tracks), K (number of APs) and C (number of MAPs).

We built three bus maps. First, we designed an irregular map, referred as *mesh*, such that to cover approximately the whole simulation area (Figure 4), trying to imitate the real bus network of a city. Later, for comparison purposes and for evaluating the influence of the coverage on the buffers and delays, we generated two regular maps as well. In the first one vertical and horizontal bus lines form a uniformly spaced (400m) *grid* (see Figure 5 left). On the other hand, the second map consists of uniformly spaced (400m), concentric octagonal bus lines as well as seven *radial* lines connecting the center to the angles of the heptagons (see Figure 5 right). This way we obtained a map (grid) with uniform coverage and another one with better coverage in its center and poorer coverage at the margins . In all three maps fixed access points were placed at every third bus stop.

Figure 6 traces the execution of a simulation run of the K2 algorithm and reports the number of messages stored in each buffer type. During the run, 1000 messages among 15 users were sent, with the urban setting shown in Figure 4 with 9 bus lines. Note that the number of sent messages steadily increases according to the Poissonian distribution, until the maximum of 1000. After this, no more messages are sent and the system stops when the last message is deleted from the last buffer. The number of delivered messages closely follows the number of sent messages. The other lines represent the number of messages stored in the various buffers involved in the delivery process. The queues of the static access points are heavily used by the algorithm, since many replicas are stored. The other queues are smaller. Note that the system reaches a “steady state” condition after a very short initial period of about 100s. This is due to the fact that the buses make an initial round before our mesasurements start. Otherwise the transient state would last about 1000s. After this initial period all experiments have shown statistical oscillations of buffer contents with roughly the same average value.

For evaluating the performance of the K2 algorithm we performed seven experiments on each map, fixing $K = 4$ and setting $C = 2, 4, 6, 8$, and vice-versa. Selected simulation results are presented in Figure 7.

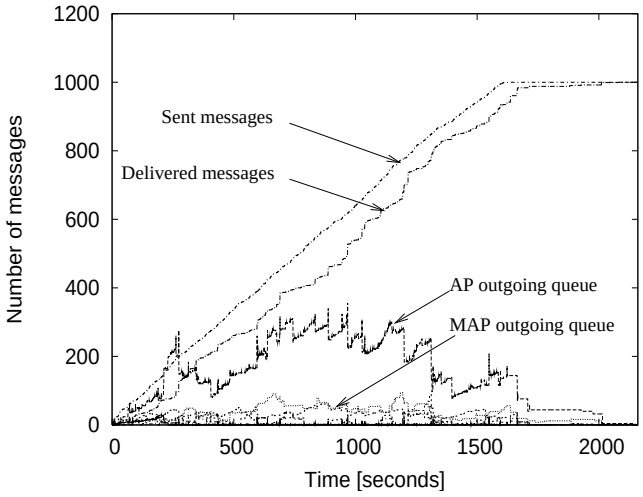


Figure 6: Trace of a simulation run

Before presenting the results in Figure 7 let us specify also the fixed settings of our simulations. During each simulation there are 15 users moving on the map generating 1000 messages. Further, we have the following number of access points generated on the different maps. On the 9 bus lines of the mesh map there are 49 APs and 24 MAPs. On the grid map there are 21 lines with 57 MAPs and 64 APs while on radial the number of lines is 22 with 49 MAPs and 42 APs. The range of the (M)APs is 200 in order to bridge in a high ratio the distances between the lines. Our simulations with a range of 100m confirmed our expectations regarding much bigger buffers and delays.

Returning to Figure 7 we can see the evolution of the most interesting parameters of the simulations. Note that in the diagrams on the left we set $C = 4$ while on the right we set $K = 4$. Further, it is good to know in advance that the average coverage of the system, computed with the Monte Carlo method, is 46.1%, 63.0% and 37.8% for the mesh, grid and radial maps, respectively.

The biggest buffers of our system are the total outgoing queues of the APs and MAPs (i.e. *APQout* and *BQout*). In Figure 7.a it can be seen that for a fixed number of MAPs, *APQout* increases as we increase the number of APs without any sign of stabilization. This is due to the fact that messages destined to users out of all fixed AP ranges will fill up the AP buffers. On contrary, when we fix K and gradually increase C the *APQout* will decrease down to a certain value, making obvious the importance of MAPs.

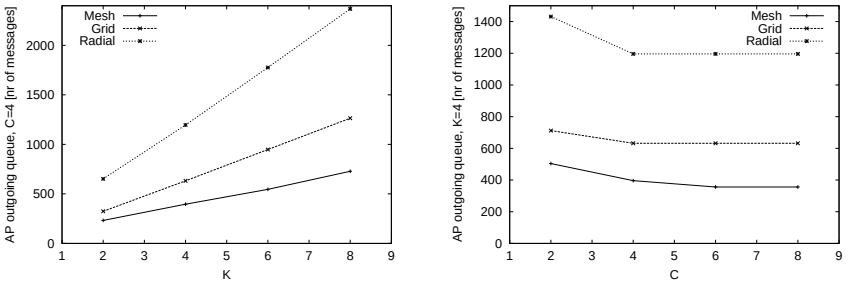
On the other hand, *BQout* will grow in both cases (Figure 7.b). The reason why *BQout* increases with K is that if the messages are available from more APs, more MAPs will receive it.

The other buffers of our system are smaller than the ones presented above but they offer scarce possibility for reduction, therefore they are of secondary importance for us. Peak values of these queues obtained with the $K2$ algorithm with K=4 and C=4 on each of our three maps are presented in Table 1.

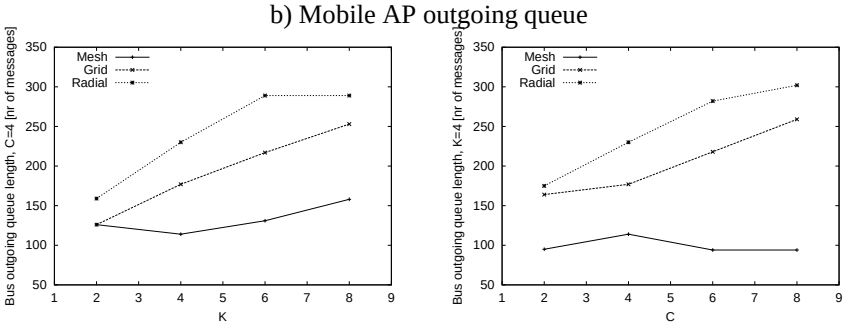
In general, user outgoing queue maximum usage is between $64 \div 121$ messages. The total MAP incoming queue is $106 \div 183$ messages big, out of which $77 \div 121$ are small acknowledgments while the sum of all AP incoming queues reaches peak values as small as $29 \div 53$ messages on the different maps.

Another set of simulations (not shown in the figure) show that using `K2` instead of `MFF` can reduce (M)AP outgoing

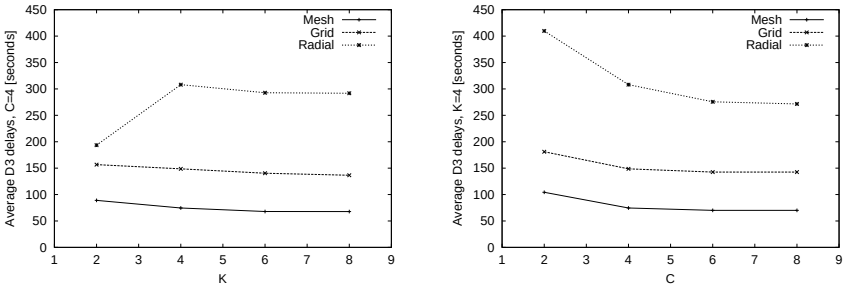
a) AP outgoing queue



b) Mobile AP outgoing queue



c) D3 delays



d) Bus-delivered messages

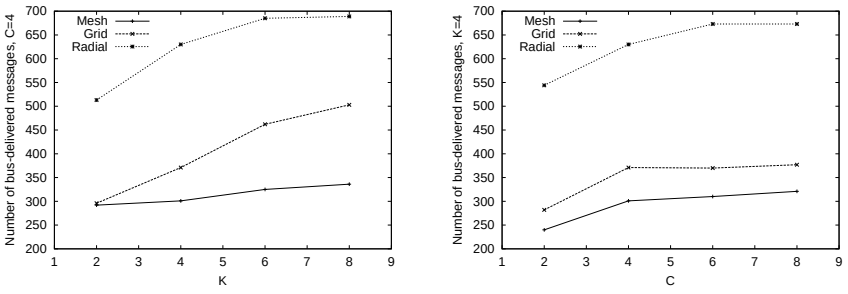


Figure 7: Disseminated simulation results for different values of K and C: a) AP buffer occupancy; b) MAP buffer occupancy; c) The most critical delay – D3; d) Number of messages delivered to destination by buses

Buffers	**Mesh**	**Grid**	**Radial**
User outgoing	64	89	121
MAP incoming	109	106	183
Acknowlegments	74	77	119
AP incoming	29	52	53

Table 1: Peak values for secondary queue usage for K=C=4

buffers by approximately $7 \div 8$ times.

Among the five kind of delays we identified that the longest one, and hence the most important, is D3 (Figure 7.c). Generally, D3 decreases as the number of (M)APs grows. The lower limit of D3 can be obtained by `MFF` since this algorithm uses all the available (M)APs. The lowest D3 we obtained (67.85s) for the above presented settings (not considering K and C) was for the mesh map. However, with only 6 MAPs and 4 APs the value of 67.93s can be obtained, using `K2`. This shows that is useless to raise the number of (M)APs above a certain limit.

Regarding the other delays, D1 and D2 remain constant during all experiments with the same map. This is due to the fact that messages are forwarded by the user to the first bus that passes in his range, while the bus will pass the message to the first AP it reaches. D4 and D5 are very small with respect to D3. D4 does not depend on K or C but on the distance between the MAP handing the message to the destination and the first AP it reaches thereafter. In change, D5 decreases with K and C since the number of messages to be deleted from the buses grows with the two constants. Sample average and peak values for these delays are shown in Table 2, where the $K2$ algorithm was run with K=C=4 on our three different maps.

Delays	**Mesh**		**Grid**		**Radial**	
	Avg	Peak	Avg	Peak	Avg	Peak
D1	51.04	530.5	88.85	856.4	118.50	1175.8
D2	15.48	111.0	47.73	293.5	75.07	450.0
D3	74.58	1189.0	148.68	2193.0	308.12	2630.5
D4	10.41	109.0	32.07	391.0	83.18	622.5
D5	2.41	116.0	6.15	214.0	30.66	436.5

Table 2: Delays for K=C=4 (average/peak value)

Finally, Figure 7.d shows that the importance of mobile access points is bigger on maps where coverage is poor, like in the case of the radial map. Indeed, this is the context for which Delay-Tolerant Networks are being designed.

7 Conclusion

In this work we presented a study on a particular DTN and we defined `K2`, a location-aware algorithm for scheduled message delivery in these networks. Further, we presented our DTN simulator that we used for evaluating buffer usage and message delivery delays while using our algorithm. We also categorized the delays in our system and after identifying the most critical one we showed how it can be manipulated through the `K2` algorithm. Finally, we obtained that `K2` can reduce buffer lengths approximately 7 times with respect to the basic `MFF` algorithm. Moreover, data on messages that were actually delivered by mobile APs to the users show that a DTN approach can be effective for many applications where extensive static coverage is unprofitable.

Although K2 produces good results when all users move with walking speed, if we diversify the moving speeds of the users, some difficulties will show up when trying to set K and C to an optimal value. Therefore, to improve K2 in the future we plan to extend it by a reactive scheme that will automatically set these parameters in concordance with the user moving speeds.

References

[1] Cecilia Mascolo, Lucia Capra, and Wolfgang Emmerich. Middleware for mobile computing (a survey). In Springer Verlang, editor, *Proceedings of the International Conference of Networking, Lecture Notes in Computer Science*, May 2002.

[2] Anders Lindgren and Olov Schelén. Infrastructured ad hoc networks. In *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02)*, Vancouver, B.C., Canada, August 2002.

[3] Matthew J. Miller, William D. List, and Nitin H. Vaidya. Hybrid network implementation to extend infrastructure reach. Technical report, University of Illinois, January 2003.

[4] Mohiuddin Ahmed, Son Dao, and Randy Katz. Positioning range extension gateways in mobile ad hoc wireless networks to improve connectivity and throughput. In *Proceedings of the IEEE Military Communications Conference (MILCOM 2001)*, Washington, D.C., USA, October 2001.

[5] Kaixin Xu and Mario Gerla. A heterogeneous routing protocol based on a new stable clustering scheme. In *Proceedings of the IEEE Military Communications Conference (MILCOM 2002)*, Anaheim, CA, October 2002.

[6] Vinton Cerf, Scott Burleigh, Adrian Hook, Leigh Torgerson, Robert Durst, Keith Scott, Kevin Fall, and Howard Weiss. Delay-tolerant network architecture. Internet draft, IETF, March 2003.

[7] Kevin Fall. A delay-tolerant network architecture for challenged internets. Technical Report IRB-TR-03-2003, Intel Research, March 2003.

[8] Xiangchuan Chen and Amy L. Murphy. Enabling disconnected transitive communication in mobile ad hoc networks. In *Proceedings of the Workshop on Principles of Mobile Computing, colocated with PODC'01*, pages 21–27, Newport, RI, USA, August 2001.

[9] Amin Vahdat and David Becker. Epidemic routing for partially-connected ad hoc networks. Technical Report CS-2000-06, Duke University, 2000.

[10] Internet Society IPN SIG. Interplanetary network project, Accessed: July 2003. Project homepage: http://www.ipnsig.org/.

[11] Avri Doria and Maria Uden. Providing connectivity to the saami nomadic community. In *Proceedings of the 2nd International Conference on Open Collaborative Design for Sustainable Innovation (DYD 02)*, Bangalore, India, December 2002.