Time Scales and Delay-Tolerant Routing Protocols

Jouni Karvo and Jörg Ott
Helsinki University of Technology (TKK)
Department of Communications and Networking
jouni.karvo@tkk.fi, jo@netlab.tkk.fi

ABSTRACT

Numerous routing protocols have been proposed for Delay Tolerant Networking. One class of routing protocols aims at optimizing the delivery performance by using knowledge of previous encounters for forecasting the future contacts to determine suitable next hops for a given packet. Protocols pursuing such an approach face a fundamental challenge of choosing the right protocol parameters and the right time scale for estimation. These, in turn, depend on the mobility characteristics of the mobile nodes which are likely to vary within one scenario and across different ones. We characterise this issue, which has been overlooked in this field so far, using PROPHET and MaxPROP as two representative routing protocols and derive mechanisms to dynamically and independently determine routing parameters in mobile nodes.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols

General Terms

Algorithms, Design, Performance

Keywords

Delay-tolerant Networking; DTN; Routing; Mobile Ad-hoc Networking; MANET

1. INTRODUCTION

Epidemic routing [9], PROPHET [8] and MaxPROP [1] are examples of routing protocols that are specialized in networks that consist only of sporadic moments of connectivity. When the nodes meet, they interchange messages that they have in their buffers, and then part again to meet other nodes.

The principle of Epidemic routing is to spread copies of data items to buffer memories of mobile devices, so that eventually some copy of the data reaches the intended destination device. This scheme allows messages to be transmitted without end-to-end connectivity. The drawback of the approach is the amount of buffer memory required in the devices, and possibly short contact durations with other nodes, limiting the data flow. To overcome these limitations, approaches that use past knowledge of encounters to optimize the spreading of data have been proposed. Such proposals include the PROPHET and the MaxPROP protocols. They estimate a "delivery predictability" or a "delivery likelihood" value, which in turn is used to decide whether a copy of the data item is forwarded to an encountered node.

The idea of estimating a predictability value for future encounters is appealing, but actually only transforms the problem of buffering and connection time limitations to the problem of finding reliable estimates for the predictability values. So far, this problem has been under-researched. Both PROPHET and MaxPROP present a scheme for estimating these, but some other proposals, such as MobySpace [7] have defined the estimation problem out of scope.

This paper shows that the estimation process can have an effect on the routing protocol behaviour. We use PROPHET and MaxPROP as examples of typical, fundamentally similar, delay-tolerant routing protocols of this class, and present a simple generalization of the MaxPROP protocol. Our main thesis is that when defining a routing protocol based on delivery predictability estimation, or simulating it, the underlying characteristics of mobility and application timescales need to be defined, in order of this type of protocol to work.

The paper is organized as follows. Section 2 describes the parameters of PROPHET, and how they affect the operation of the protocol. Section 3 discusses the MaxPROP protocol mechanism similarly, presenting a simple generalization to the protocol. We describe the approach of defining protocol parameters starting from the desired timescale for the protocol to work in Section 4. Section 5 shows first simulation results for PROPHET using different parameter settings. Finally, Section 6 concludes the paper.

2. PROPHET

PROPHET [8] is a DTN routing protocol aiming at using knowledge obtained from past encounters with other nodes to optimize the packet delivery. Each node keeps a vector of *delivery predictability* estimates, and uses it to decide whether an encountered node were a better carrier for a DTN packet.

The predictability estimates are increased every time a node encounters another node, and they are decayed exponentially. This process is treated in more detail in the following subsections. The PROPHET protocol also includes a "transitivity" mechanism (controlled by a parameter β) for dealing with the case where two nodes rarely meet, but there is another node that frequently meets both of these nodes. The updating mechanism of the transitivity property is not needed for the purpose of this paper, so it is left for future work. I.e. assume $\beta = 0$.

2.1 Predictability estimation parameters

The PROPHET specification defines two aging parameters: $\gamma \in (0,1)$ as the *aging constant*, and *time unit*, u, defined based on the application. These constants are actually dependent, and there is only one free parameter.

Let t denote the elapsed time since the aging process started (i.e. after the last delivery predictability calculation). Let P_{orig} denote the predictability at the time the aging started. Define a continuous function

$$P_t = P_{\text{orig}} e^{-bt},$$

where b is a constant. Equation (2) in [8] defines aging as

$$P_{(a,b)} = P_{\text{orig}} \gamma^k,$$

where k is the number of time units since the last aging update. Assuming that at the discrete moments of time $t = ku$, $P_t \overset{\text{def}}{=} P_{(a,b)}$. Then,

$$\begin{aligned}
P_{\text{orig}} e^{-bt} &= P_{\text{orig}} \gamma^k \\
e^{-bt} &= \gamma^k && |P_{\text{orig}} \neq 0 \\
\left(e^{-b}\right)^t &= \left(\gamma^{\frac{1}{u}}\right)^t = \left(e^{\frac{1}{u}\ln\gamma}\right)^t \\
e^{-b} &= e^{\frac{1}{u}\ln\gamma} && |t \neq 0 \\
b &= -\frac{1}{u}\ln\gamma. && (1)
\end{aligned}$$

Since there is in practice only one aging parameter, b, the choice of the time unit can be arbitrary, and be based on technical considerations, such as how to measure time in a specific device. The parameter γ can then be used to achieve the desired b.

PROPHET formulates the update of the predictability, ignoring transitivity definition, as

$$P_{(a,b)} = P_{(a,b)\text{old}} + (1 - P_{(a,b)\text{old}}) P_{\text{init}}. \tag{2}$$

Define P' as $1 - P_{(a,b)}$. Then,

$$P'_{\text{new}} = P' - P' P_{\text{init}} = (1 - P_{\text{init}}) P'.$$

In case a node pair meets with constant intervals, then after n meeting times,

$$P'_n = (1 - P_{\text{init}})^n P'.$$

We see that the formulation follows the same exponential form as for the decay of the predictability. Let us write (2) using $\zeta = 1 - P_{\text{init}}$:

$$P_{(a,b)} = 1 - \zeta^n \left(1 - P_{(a,b)\text{original}}\right).$$

Rewriting the equation in continuous time, as for the decay,

$$P_{(a,b)} = 1 - e^{-ct} \left(1 - P_{(a,b)\text{original}}\right).$$

Here,

$$c = -\frac{1}{u'} \ln\zeta = -\frac{1}{u'} \ln(1 - P_{\text{init}}), \tag{3}$$

where u' can be interpreted as a time unit for increasing predictability. We see that P_{init} is actually a constant defining the rate of increase for predictability. For reducing confusion, we use $\zeta = 1 - P_{\text{init}}$ in the following sections.

This formulation also reveals a natural way of extending the delivery predictability calculation of PROPHET. If the contact lengths are considered to also matter for delivery predictability, so that nodes that tend to have long contact times with each other have bigger delivery predictability, PROPHET can be changed so that delivery predictability is updated by e^{-ct} during a contact, instead of ζ for each contact. To convert between ζ and c, set u' to the mean contact duration.

2.2 Operation of the predictability estimator

In order to get an idea of what parameter values can be used, and how the protocol works, first consider a system consisting only of two nodes. The system oscillates between two states: First, the predictability value is increased, as the nodes meet each other. Then, the predictability value is decreased during a period of length B, until the next encounter. The system starts from a value P_0 at time $T = 0$. To relate these parameters to the "real world" protocol operation, set B to the sum of average contact and intercontact-time, or the average time between encounters of node pairs. We call this *interencounter time*. In the beginning of the cycle, an encounter happens, and the predictability value reaches the maximum:

$$P_{T=0+} = P_{\max} = 1 - \zeta(1 - P_0). \tag{4}$$

And at time $T = B$, the predictability value reaches its minimum again:

$$\begin{aligned}
P_{T=B} &= P_{\max} e^{-bB} \\
&= (1 - \zeta(1 - P_0)) e^{-bB}.
\end{aligned}$$

When the cycle repeats, the predictability estimate changes, periodically increasing and decreasing. Consider the points P_n, which denote the predictability value in the end of cycle n. First,

$$\begin{aligned}
P_n &= (1 - \zeta(1 - P_{n-1})) e^{-bB} \\
&= \underbrace{e^{-bB}(1-\zeta)}_{\overset{\text{def}}{=} C} + P_{n-1}\zeta e^{-bB} \\
&= C + P_{n-1}\zeta e^{-bB} \\
&= C + \left(C + P_{n-2}\zeta e^{-bB}\right)\zeta e^{-bB} \\
&= C\left(1 + \zeta e^{-bB}\right) + P_{n-2}\left(\zeta e^{-bB}\right)^2 \\
&= e^{-bB}(1-\zeta)\sum_{i=0}^{n-1}\left(\zeta e^{-bB}\right)^i + P_0\left(\zeta e^{-bB}\right)^n \\
&= \frac{1-\zeta}{e^{bB}-\zeta}\left[1 - \left(\zeta e^{-bB}\right)^{n-1}\right] + P_0\left(\zeta e^{-bB}\right)^n && (5)
\end{aligned}$$

In this kind of a periodic system, the predictability value stabilizes to oscillate around the same values, and $\lim_{n\to\infty} P_n = P_{\text{stable}}$. It is sufficient for the predictability to stabilize, when $bB > 0 \Rightarrow \gamma < 1$, since $\zeta < 1$. Then, the system converges to

$$\begin{aligned}
\lim_{n\to\infty} P_n &= \frac{1-\zeta}{e^{bB}-\zeta}\left[1 - \underbrace{\left(\zeta e^{-bB}\right)^{n-1}}_{\to 0}\right] + P_0\underbrace{\left(\zeta e^{-bB}\right)^n}_{\to 0} \\
&= \frac{1-\zeta}{e^{bB}-\zeta} = P_{\text{stable}}.
\end{aligned}$$

When the system has reached its stable state, the average predictability $\overline{P}$ is

$$\begin{aligned}
\overline{P} &= \frac{1}{B}\int_0^B P_{\max} e^{-bt}\,dt \\
&= \frac{1 - e^{-bB}}{bB}\left(1 - \zeta(1 - P_{\text{stable}})\right).
\end{aligned}$$

There are two main choices for the PROPHET parametrization, illustrated in Figures 1 and 2. The first choice of parameters aims at estimating the predictability as a longer time average, while the second aims at a short term effect. The time that is needed for reaching a steady state needs to be taken into account also in the simulations, especially with the first approach. Considering the parameters in Figure 1, for example, a suitable warm-up period is in the order of 30 cycles, meaning the node pair has time to meet 30 times before the predictability value is on approximately the right level. If the values are interpreted as seconds, i.e. nodes encounter each other on average every 2010 seconds for 10 seconds, a bit over half an hour between each encounter, the simulation warmup time should be around 17 hours of simulation time. Or, in a real-life scenario, it would take the corresponding time before the protocol starts to work as intended.

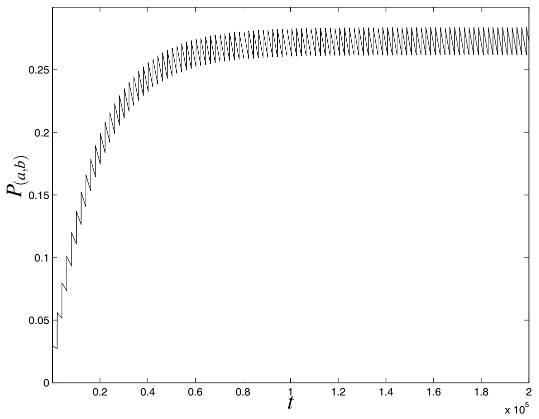


Figure 1: PROPHET reaching the steady state. $B = 2000$**,** $\zeta = 0.9704$**,** $b = 4 \cdot 10^{-5}$**. Resulting** $P_{\text{stable}} = 0.2622$**,** $\overline{P} = 0.273$ **and** $P_{\max} = 0.284$**.**

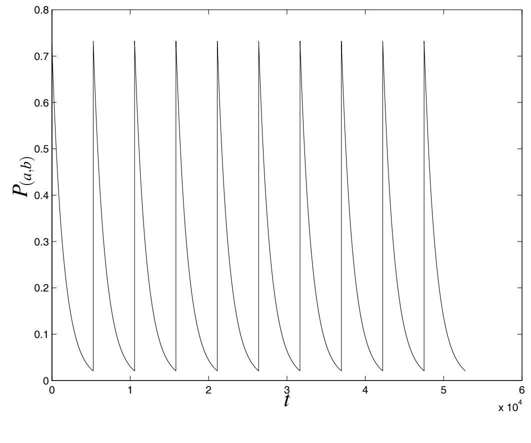


Figure 2: Predictability with the default settings for PROPHET routing in the ONE simulator. $B = 5274$**,** $\zeta = 0.2737$**,** $b = 6.7333 \cdot 10^{-4}$**. Resulting** $P_{\text{stable}} = 0.021$**,** $\overline{P} = 0.2002$ **and** $P_{\max} = 0.7320$**.**

Figure 2 shows the behaviour of PROPHET with the default parameters of the ONE simulator [6], but with transitivity turned off ($\beta = 0$). This shows the second possible mode of the PROPHET algorithm, where a single encounter is able to set the predictability to a high value, which is then quickly reduced. Using this mode assumes that it is most probable to meet again the nodes that the node just met. The longer it has passed after an encounter, the more unlikely it is that the nodes meet again. Using this approach, there is no cumulative evidence of nodes having met several times before. This mode of operation is based on the assumption that the intercontact time distribution is heavy tailed—the longer it has taken from the last encounter, the longer it will take until the next encounter. There have been such results available [2, 5], so both of these modes of PROPHET usage are possible, depending on the scenario. As regards to the warmup period of the protocol with this mode of operation, one cycle of contact time and intercontact time is enough.

3. MAXPROP

MaxPROP [1] is another protocol using knowledge from previous encounters for making delivery optimization in DTN. The idea is that each node keeps a vector called *delivery likelihood*, which is obtained using incremental averaging. When two nodes meet, they exchange these vectors, and so each node can calculate the shortest path to the destination.

The algorithm is based on calculating the likelihood of connecting next to node j. Each node has a vector $F^i = (f^i_0 \dots f^i_{J-1})$, normalized so that the sum of the elements is 1. When a node encounters another node, the element of the vector corresponding to the node encountered is incremented by 1, and then all vector elements are divided by 2. This way, the vector is kept normalized all the time.

When calculating these values in practice, the node that was met last has always the highest value, and the node met before that a value approximately half of that value, and so on. This way, the likelihood vector can be seen as an ordering of the nodes. Whenever the node meets the next node, the newly encountered node is put on the start of this order. This definition of likelihood is extremely volatile. The shortest path mechanism uses these volatile likelihood vectors for path cost calculations. This is a problem, since the earlier exchanged likelihood vectors are already obsolete when the calculations are done.

In this sense, MaxPROP resembles a bit the PROPHET routing algorithm in the second mode, where the last encounter is considered most important, and history of past encounters is considered less significant. The difference is in the aging process; while PROPHET uses time to age the predictability values, in MaxPROP, a node forgets past encounters when encountering new nodes.

We generalize MaxPROP so that the averaging increment $\alpha > 0$ can be arbitrarily chosen, instead of the default $\alpha = 1$. The protocol can then use the history of past encounters as additional evidence when calculating delivery likelihood. This leads to a kind of a sliding average of the frequency of meetings of other nodes, resembling the first mode of PROPHET algorithm. Selecting the correct value of α depends on what is the desired time scale for forgetting about older contacts, and depends on the movement model and the application.

3.1 MaxPROP behaviour

Figure 3 shows an example of the default behaviour of the MaxPROP delivery likelihood estimate. The node meets other (non-interesting) nodes every 20 time units, and one node every 100 time units. The delivery likelihood of the latter is shown in the figure. Figure 4 shows the same scenario but with a modified MaxPROP, with $\alpha = 0.1$ used as the increment for incremental averaging. As a result, the likelihood sets after an initial transient to a level indicating the relative frequency of encounters. As a penalty, it takes some time in the beginning (in this case, approximately 5 cycles) before the likelihood level is reached.

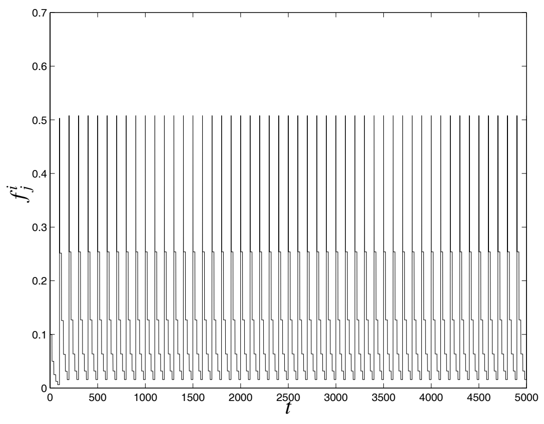


Figure 3: MaxPROP likelihood behaviour with default likelihood update.

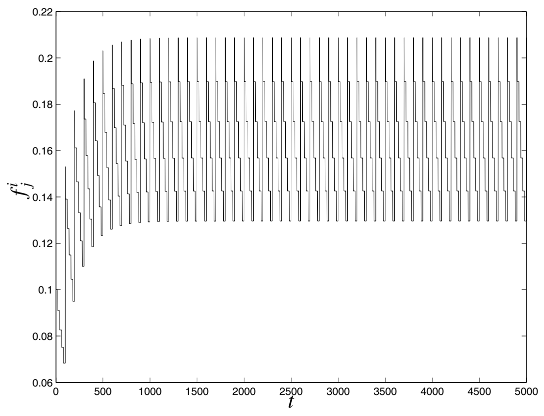


Figure 4: Modified MaxPROP likelihood behaviour with $\alpha = 0.1$.

To be more precise, we define that the delivery likelihood is updated as follows. Whenever an encounter with node j happens,

$$f^i_{j,n+1} = \frac{f^i_{j,n} + \alpha}{1+\alpha},$$

and when a node encounters a node different than j,

$$f^i_{j,n+1} = \frac{f^i_{j,n}}{1+\alpha}.$$

Consider a cycle, where a node encounters L other nodes in between each encounter with node j. The cycle starts with $f^i_j = f_0$. After the cycle, f^i_j is

$$f_1 = \frac{\frac{f_0+\alpha}{1+\alpha}}{(1+\alpha)^L} = \frac{f_0+\alpha}{(1+\alpha)^{L+1}}.$$

Assuming the same cycle repeats, f_n is then

$$f_n = \frac{\alpha}{(1+\alpha)^{L+1}-1}\left[1-(1-\alpha)^{-n(L+1)}\right] + \frac{f_0}{(1+\alpha)^{n(L+1)}}.$$

Note that this equation is of the same form as (5). The second term diminishes as cycles repeat. When $\alpha > 0$, the first term converges, as $(1+\alpha)^{-(L+1)} < 1$. The process stabilizes at

$$\lim_{n\to\infty} f_n = f_\infty = \frac{\alpha}{(1+\alpha)^{L+1}-1}. \quad (6)$$

Filling in the parameters of the scenarios for Figures 3 and 4, for $\alpha = 1$, $f_\infty \approx 0.0129$ and for $\alpha = 0.1$, $f_\infty \approx 0.130$.

4. DEFINING PROTOCOL PARAMETERS BASED ON TIME SCALES

So far we have seen that protocol behaviour can change drastically, depending on how the protocol parameters are chosen. The obvious question is then: How should the protocol parameters be chosen?

We claim that the correct way to define protocol parameters is starting with the scenario and applications. The expected contact and intercontact times, and numbers of encounters typically depend on the day and time of day. In some applications, where the DTN messages age quickly, only short term behaviour is interesting. Then, choosing parameters that cause the algorithms to stabilize quickly are chosen. For other applications, longer term adaptation can be appropriate. Consider for example email, where delivery the next day is good enough — the predictability or delivery likelihood values need to reflect the daily occurring encounters, instead of the last minutes, in order to benefit from daily routines.

Thus, the essential question is the time scale. Time scale is defined here as the time it takes for the system's predictability or likelihood values to adapt to a different mobility pattern. For example, a time scale of 1 hour means that when a change in mobility patterns occurs, it takes approximately one hour for the predictability or delivery likelihood values to reach the new level.

To turn this idea into practice, we use the models prepared in the previous sections to find parameters for both PROPHET and MaxPROP to be used in more realistic scenarios.

4.1 PROPHET parametrization

Let T denote the desired time scale, and B the mean interencounter time. Then, the number of cycles n_{target} corresponding to this target time scale,

$$n_{\text{target}} = \left\lceil \frac{T}{B} \right\rceil, \quad (7)$$

can be used to get an idea whether the chosen PROPHET parameters b and ζ are meaningful. For this, choose b and ζ so that for all P_0 they create the maximum error smaller than or equal to ε,

$$\left|P_{n_{\text{target}}} - P_{\text{stable}}\right| \leq \varepsilon, \quad (8)$$

where the error ε is a chosen suitably small constant, for example 0.1. We note that even fixing the time scale, and knowing the mean interencounter time, there are two parameters that can be chosen. Setting the average predictability value to an arbitrary constant, such as $\overline{P} = 0.2$, can be used to fix both of the parameters. This lets the predictability values some space to fluctuate depending on the actual number of encounters for node pairs.

Different devices will see different types of usage; some devices might stay at people's homes most of the time, while some other devices see many encounters during the day. Since the protocol behaviour depends on the frequency of encounters, having a single set of parameters for the devices can be suboptimal.

The desired timescale of the protocol needs to be decided based on the time to live of the application bundles. For example, a 4 h and 12 day time-to-live settings required different time scales also for the application.

To cover the different usage of different devices, an algorithm can be devised for estimating the mean interencounter times, and then used for finding the parameters of the PROPHET. In case there is no trend component in traffic, a simple mean of interencounter times suffices:

$$\widehat{B}_n = \frac{n-1}{n}\widehat{B}_{n-1} + \frac{1}{n}e_n,$$

where e_n denotes the nth measured interencounter time. These times need to be estimated over all node pairs separately, but they are used to calculate a single mean. In the case where there is a trend, another timescale problem emerges; the estimation needs to be implemented in a way that gives more weight to the newest measurements in a controlled way.

Given the estimated $\widehat{B}$, and other parameters, e.g. $\varepsilon = 0.001$, $\overline{P} = 0.2$, and the target timescale T, calculating the PROPHET parameters can be done with two nested iterative steps:

1. The outer loop is for searching ζ:

   - If $\overline{P} > \overline{P}_{\text{target}}$, increase ζ
   - If $\overline{P} < \overline{P}_{\text{target}}(1-\varepsilon)$, decrease ζ
   - otherwise: parameters are found

   The amount of increase or decrease is reduced when the iteration advances.

2. The inner loop is for finding b for the chosen ζ

   - If the error from (8) after the target timescale is bigger than ε, increase b (PROPHET does not converge quickly enough).
   - If the error (8) is smaller than $\varepsilon(1-\varepsilon)$, decrease b (delivery predictability converges too quickly).

   The amount of increase or decrease is reduced when the iteration advances. If the inner loop is not able to find a suitable b, increase ζ, and try again.

The algorithm will find the parameters of PROPHET in a reasonable amount of time. The requirement for this algorithm to work is that the target timescale is longer than the mean interencounter time. If it is smaller, PROPHET works in the mode shown in Figure 2.

We simulated PROPHET delivery predictability with Poisson distributed encounters with the intensity modulated by a sine wave, for showing a daily variation of interencounter times. Figure 5 shows the delivery predictability for an application with target timescale of 1 hour, in a case where the node pair meets on average every 16.7 minutes. The node pairs meet only during the day, and since the application timescale is only one hour, the delivery predictability diminishes during the night.

Figure 6 shows a simulation with the target timescale of about 30 days, and with mean interencounter time of 8 hours. The results show that finding the correct average level of delivery predictability takes a long time, but on the other hand, the daily variation in encountering intensity does not affect delivery predictability as heavily as with a lower timescale. The implementation of the protocol requires estimation of the mean interencounter time. This requires storing the time of the last encounter for each node. Each time a new encounter happens, the interencounter time can then be calculated easily, and the mean calculated from all node pair's interencounter times can be updated.

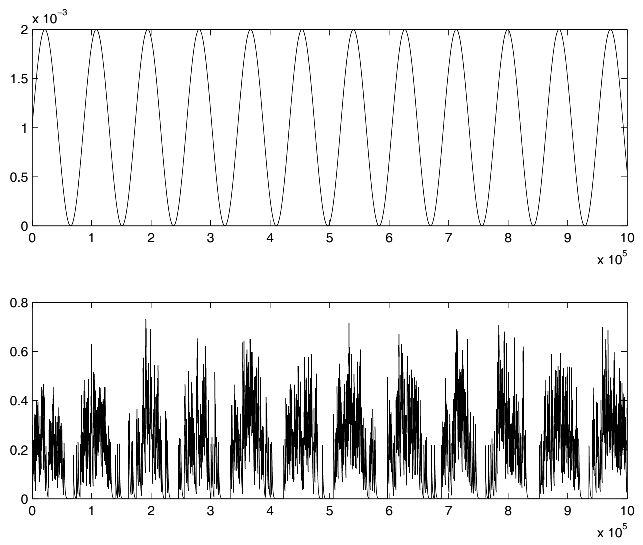


Figure 5: PROPHET with parameter estimation. The target timescale $T = 3600$ s. Mean interencounter time: 16.7 min. Top: the encountering intensity, bottom: estimated delivery predictability.

4.2 MaxPROP parametrization

MaxPROP is defined with the encounters, and thus when parametrization starts from the time scale T, the needed parameters are the mean interencounter time B and the mean number of encounters between contacts of a specific node pair L.

The number of cycles n_{target} is again defined as in (7). Then, the parameter α for MaxPROP can be defined in a similar way as for the PROPHET, from maximising the error for all f_0:

$$\left| f_{n_{\text{target}}} - f_\infty \right| \leq \varepsilon,$$

where ε is a suitable error constant. For MaxPROP, the choice of parameter α also affects the level on which the delivery likelihoods fluctuate, but there is no way of controlling that level.

MaxPROP can be tuned in the same way as PROPHET to work automatically on a specific timescale, but solving the parameter α for MaxPROP is significantly easier to do than solving the correct parameters for PROPHET. The nodes need to estimate two parameters in this case, though. First, mean interencounter time $\widehat{B}$ needs to be estimated, as for PROPHET. Second, the mean number of contacts during an interencounter time $\widehat{L}$ needs to be estimated. The estimation of L can be done easily by having a counter for encounters, and storing the counter value of the last encounter for each encountered node. When the nodes meet again, it is straightforward to calculate the number of encounters with other nodes in between, and thus updating the estimate $\widehat{L}$.

With these estimates, and the target timescale, the MaxPROP timescale parameter α can be estimated iteratively, in the same way as the two parameters of PROPHET. In this case, though, only a single loop is required, instead of the two nested ones in PROPHET.

5. SIMULATIONS AND DISCUSSION

5.1 Idealistic viewpoint

Figure 7 illustrates the expected behaviour of PROPHET as a function of message TTL. The two curves show message delivery ratio for two parametrizations, calculated from different time scale

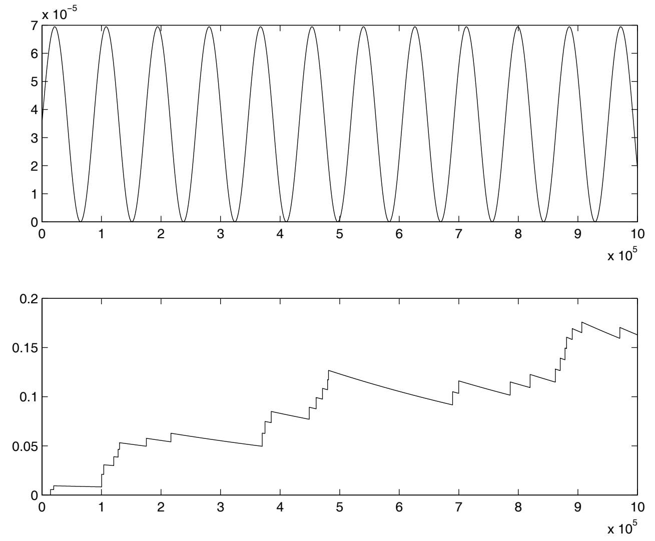


Figure 6: PROPHET with parameter estimation. The target timescale $T = 2.6 \cdot 10^6$ s (≈ 30 days). Mean interencounter time: 8 h. Top: the encountering intensity, bottom: estimated delivery predictability.

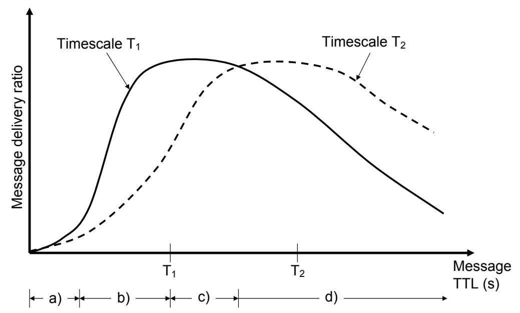


Figure 7: Qualitative outline of the expected behavior

targets. There are four identifiable ranges shown for curve T_1:

a) If the message TTL is less than the mean interencounter time, the routing and forwarding process is essentially reduced to random delivery. Any routing protocol behaves badly, as the nodes do not meet often enough for the message to be able to reach the destination. In this case, the routing essentially degenerates to the *direct delivery* or *first contact* [4].

b) If the message TTL is larger than the interencounter time but less than the timescale, history information starts getting useful. The message may on average not be able to travel many hops, but could reach the destination. Since the history information is collected from a significantly longer period of time than the message TTL, it can contain information of nodes that encountering less frequently, incurring an error.

c) If the message TTL is in the order of the timescale, history information can be best used for forwarding—the delivery predictability values are estimated on the time scale within a message's TTL. As an additional note, we expect the performance on this range to be highest when the time scale is sufficiently larger than the interencounter time, so that there are more samples used for the estimation.

d) If the TTL is larger than the timescale, the protocol performance starts to degrade, as information on node pairs that meet with longer interencounter times do not get recorded properly. As a result, suboptimal paths can be chosen, so that we expect the message delivery ratio to decrease again.

In summary, there are two important aspects: 1) The routing protocols run by nodes need to get enough time to build up meaningful state for the environment they are operating in. Depending on the convergence time, the results will be biased before some stable state is reached. 2) The lifetime of messages sent in a particular environment needs to match the timescales in the system: i.e., the right routing needs to be picked.

5.2 Simulated results

We run simulations for both PROPHET and MaxPROP to obtain a first more realistic validation of our routing protocol parameterization: We used the *Working Day Movement Model* [3]. We choose the default scenario from section 5 in [3] in which nodes move in the Helsinki city area, but reduce the number of nodes from 1 029 to 544 nodes by shrinking all the group sizes so that the basic contact characteristics remain. We use half a day of warmup time for the mobility model so that the nodes can spread before starting transmitting messages.

In our simulations, the nodes transmit messages with the message sizes uniformly distributed between 1 KB and 1 MB and each node has 100 MB of buffer space. For message transmission, the simulator draws intervals from a uniform distribution. After each interval a (sender, receiver) node pair is drawn, a message sent, and a new interval is drawn from the uniform distribution. For the basic offered load (denoted as "1x"), each of 500 nodes sends one message per day on average. We also run simulations with twice (2x), four (4x) and eight times (8x) the basic load to assess the impact of different offered loads. The active message generation period is 1 day (86 400 s) to which we add a warmup and cooldown time to initialize the nodes buffers to a steady state and allow for the delivery of already sent messages, respectively.

Warmup and cooldown periods are equal to the message TTL, with a minimal warmup of 100 000 s to allow for some contact history to build up. We run simulations with different constant message TTLs: 10 000 s, 100 000 s, 200 000 s, 300 000 s, and 600 000 s, to assess the impact of time scales on messages with different application requirements. The total simulation times are thus 196 400 s, 286 400 s, 486 400 s, 686 400 s, and 1 286 400 s, respectively.[1] We use the packet delivery ratio as the performance indicator. The measured values are means over five simulation runs for PROPHET and over three simulation runs for MaxPROP.

We first run the simulator and the movement model in order to measure the mean interencounter times B and the mean number of contacts between encounters L in our simulation scenario. Since the durations for the various simulations differ, so will the observed interencounter times and contacts between encounters (longer simulation times allow for larger interencounter times), see Table 1.

We can determine the PROPHET parameters P_{init} and γ as well as α for MaxPROP for different time scales for each simulation scenario, using the measured B and L. Table 2 shows, as an example, the parameter values found for the simulation setup with TTL=100 000 s. It is apparent that they differ significantly from the respective default parameters. This holds also for the other message TTLs.

[1]The base configuration for the ONE simulator including the mobility parameters and the message events is available from http://www.netlab.tkk.fi/tutkimus/dtn/sim/2008-09-timescales.tar.

Table 1: Mean interencounter times B and number of contacts between encounters L for different simulation durations.

Msg TTL	Duration	B	L
100 000 s	286 400 s	49 051 s	173
200 000 s	486 400 s	82 813 s	251
300 000 s	696 400 s	111 145 s	306
600 000 s	1 286 400 s	197 522 s	445

Table 2: Parameter values for PROPHET and MaxPROP for simulations with a message TTL=100 000 s: Original and values determined for different time scales.

Time scale	PROPHET		MaxPROP
	P_{init}	γ	α
Original	0.75	0.98	1.0000
100 000 s	0.2736	0.99997493	0.0089
200 000 s	0.1789	0.99998417	0.0053
300 000 s	0.1324	0.99998851	0.0038

We have implemented two modified versions of each PROPHET and MaxPROP: one which can be parameterized according to the common average values shown in Table 2 before starting the simulations ("predetermined estimation") and one which performs the parameter estimation as a continuous process independently in each node during the simulations, using the algorithms outlined in Section 4 ("dynamic estimation"). The dynamic estimation code presented is sufficiently lightweight and fast that it can be run on mobile nodes once in a while (e.g., per encounter).

For predetermined parameter estimation (not shown), we find a minor improvement of the message delivery ratio when using PROPHET routing with $\beta = 0$. While only minor (1–5% for the delivery ratio), we emphasize that we do not use node-specific parameterization, but only common values for all nodes based upon the mean calculated across all nodes. Therefore, we care more about the qualitative observation that intentional parameterization *can enable* improvements in the first place.

The impact of dynamic parameter estimation on the message delivery ratio of PROPHET routing (again with $\beta = 0$) are shown in Figure 8 comparing the original PROPHET to the modified one for different timescales and offered loads. We find a similar performance for small TTLs and a slight performance decrease for larger ones compared to the original PROPHET. We attribute this to the fact that the dynamic parameter estimation has to learn the contact characteristics over time and will have incomplete information in the beginning, resulting in incorrect estimates. With increasing message TTL, long interencounter times gain relevance, but these are naturally observed less frequently, leading to suboptimal estimates. We also observed that the relative performance is largely independent of the offered load.

These findings may suggest that meaningful default values should be chosen as long as no sufficiently solid estimate can be provided from the observed encounters for a given message TTL.

As most parameters can be estimated, the question remains which message lifetimes a routing protocol should assume: the applications running on a mobile node may be too different. This calls for categorizing application messages into TTL classes (similar to QoS classes in Internet forwarding) and running independently routing protocol parameterizations for these—and, for very short-lived messages, possibly even different routing protocols.

While we would also expect a correlation between the observed performance and the agreement between the message TTL and the chosen timescale, these do not emerge. We believe that are timescales

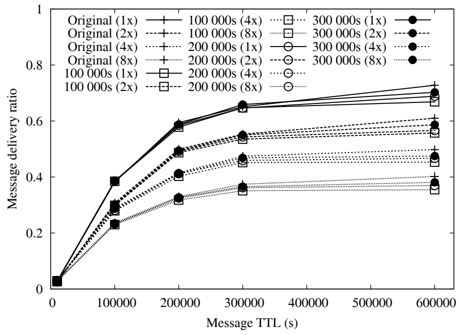


Figure 8: Message delivery ratio and delivery latency as a function of message TTL for different offered loads: Original PROPHET parametrization, and three time scale defined parametrizations.

are too close and that differences in a order of magnitude or more are needed to show this effect. This is confirmed by the original PROPHET curve showing the same shape as the ones adapted to time scales, which indicates that the time a message may be buffered and carried around is still the dominating factor.

5.3 Transitivity

So far we have omitted the effect of transitivity in the analysis. PROPHET defines transitivity as

$$P_{(a,c)} = P_{(a,c)_{old}} + (1 - P_{(a,c)_{old}}) P_{(a,b)} P_{(b,c)} \beta.$$

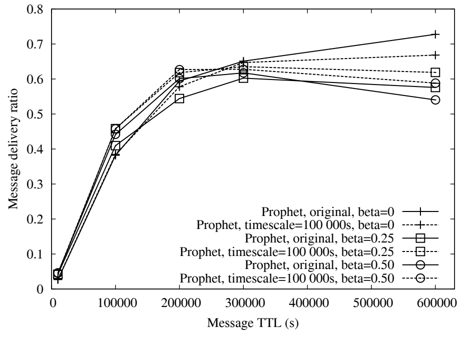
We notice that with this formulation, the delivery predictability values are mixed together to form a kind of "path" values, not keeping track of the individual "link" values. Figure 9 shows the effect of the transitivity parameter β. The default parameter set of PROPHET works as in Figure 2 — it uses highly volatile values for delivery predictability. This works well with $\beta = 0$, since the beginning of the interencounter time distribution for the used mobility model follows a power-law. Thus, the time elapsed from the last encounter with another node can be used to predict the time left until the next encounter. This does not seem to hold for multi-hop paths, though: instead, the delivery predictability values for the next hops add as random noise to the delivery predictability values. The effect is more pronounced with higher load.

The parametrization based on time scales aims at PROPHET working in a more stable form, as in Figure 1, and thus the delivery predictability values approximate the intensity of meeting other nodes. Summing these values together appears to produce less random noise, and thus the protocol performs better with $\beta = 0.25$ than with the default parameters.

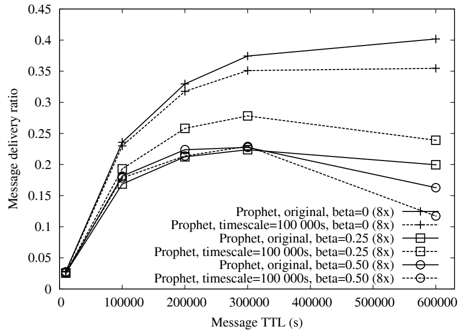
Unlike PROPHET, MaxPROP separates link-level delivery likelihood values and path costs. The past contacts between nodes are used to create a virtual topology of the network, and each path through the network has a cost, calculated as

$$c(i, i+1, \ldots, d) = \sum_{x=i}^{d-1} [1 - f_{x+1}^{x}].$$

This approach has a nice appeal, since it is a well-defined shortest path calculation, but in practice the normalization of the delivery likelihood values results in all link costs being near 1, thus reducing the algorithm to a shortest hop count metric. The delivery likelihood values are on average $1/\delta$ where δ is the degree of the node. Consider an example with $\alpha = 1$. In the extreme case where a node meets only two nodes ever, the link costs alternate between $1/3$



(a) Traffic load: 1 message / node / day



(b) Traffic load: 8 messages / node / day

Figure 9: The effect of β on the PROPHET delivery ratio

and 2/3. For any node with more connections, the link costs are approximately the values from the series 0.5, 0.75, 0.875, 0.9375, ... This suggests that the choice of cost metric slightly favours nodes that do not meet many other nodes, but it is unclear if the effect is significant. Figure 10 confirms that the protocol function also in practice discards the delivery likelihood values; irrespective to the parametrization, the results are identical.

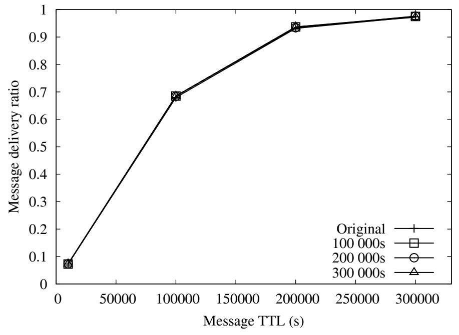


Figure 10: The effect of parametrization on the MaxPROP delivery ratio

6. CONCLUSIONS

This paper analyzed the impact of timescales on DTN routing protocols. Our theoretical considerations have used two representative DTN routing protocols: PROPHET and MaxPROP that update "predictability" information in a way that timescales matter. We have shown in simple simulation models that the proper parameterization matters and partly confirmed this for a reasonably large-scale real-world scenario focusing on PROPHET. We also found that multi-hop paths are unsatisfactorily treated in the protocols: for PROPHET, using the transitivity parameter reduced packet delivery rate, and MaxPROP discarded the delivery likelihood information with the path calculation. Thus, the treating of multi-hop path routing is still an open research issue.

While we have provided a first analysis of the impact of timescales on DTN routing protocols, there is a wealth of aspects still open for future research. We have to understand how different mobility characteristics impact the estimation and the resulting performance and explore using further mobility models. For performance measurements, we have yet to isolate the impact of the interaction between the buffer space and message TTL. And for generalizing to further routing schemes, we need to compare single-copy and multi-copy approaches and investigate the effect of varying the number of message copies for the latter.

Since our algorithms for dynamically obtaining routing protocol parameters can be run on individual mobile nodes, we need to understand how these parameters evolve over time and how diversity across different nodes affects punctual as well as overall system performance. Such diversity implies that some common assumptions about the system may no longer hold and thus require additions to the routing signaling to explicitly communicate the timescale may be required. We expect this at least, when we introduce multiple TTL classes to routing protocols to match different application requirements. Finally, besides tailoring routing protocols to different application requirements, we are interested in further exploring how to best address the very short-lived messages.

7. REFERENCES

[1] BURGESS, J., GALLAGHER, B., JENSEN, D., AND LEVINE, B. N. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In *Proc. IEEE INFOCOM* (April 2006).

[2] CHAINTREAU, A., HUI, P., CROWCROFT, J., DIOT, C., GASS, R., AND SCOTT, J. Impact of Human Mobility on the Design of Opportunistic Forwarding Algorithms. In *Proc. IEEE INFOCOM* (April 2006).

[3] EKMAN, F., KERÄNEN, A., KARVO, J., AND OTT, J. Working day movement model. In *Proc. 1st ACM/SIGMOBILE Workshop on Mobility Models for Networking Research* (May 2008).

[4] JAIN, S., FALL, K., AND PATRA, R. Routing in a Delay Tolerant Network. In *Proceedings of the ACM SIGCOMM* (September 2004).

[5] KARAGIANNIS, T., BOUDEC, J.-Y. L., AND VOJNOVIC, M. Power law and exponential decay of inter contact times between mobile devices. In *Proc. ACM MobiCom* (September 2007), E. Kranakis, J. C. Hou, and R. Ramanathan, Eds., pp. 183–194.

[6] KERÄNEN, A., AND OTT, J. Increasing reality for DTN protocol simulations. Technical Report, Helsinki University of Technology, Networking Laboratory, July 2007.

[7] LEGUAY, J., FRIEDMAN, T., AND CONAN, V. Evaluating mobyspace-based routing strategies in delay-tolerant networks: Research articles. *Wirel. Commun. Mob. Comput. 7*, 10 (2007), 1171–1182.

[8] LINDGREN, A., DORIA, A., AND SCHELÉN, O. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev. 7*, 3 (2003), 19–20.

[9] VAHDAT, A., AND BECKER, D. Epidemic routing for partially-connected ad hoc networks. Tech. Rep. CS-2000006, Duke University, April 2000.