Research Journal of Applied Sciences 7 (4): 234-239, 2012
ISSN: 1815-932X

Reducing End to End Delay in Delay Tolerant Mobile Network Using AODV and Cluster-Based Routing Protocol

S. Kannan, V.P. Arunachalam and S. Karthik
SNS College of Technology, Coimbatore, Tamilnadu, India

Abstract: This research investigates distributed clustering scheme and proposes an Ad hoc On demand Distance Vector (AODV) routing protocol for Delay-Tolerant Mobile Networks (DTMNs). The basic idea is to distributively group mobile nodes with similar mobility pattern into a cluster which can then interchangeably share their resources (such as buffer space) for overhead reduction and load balancing, aiming to achieve efficient and scalable routing in DTMN. An ad hoc network is self-organizing and adaptive. Networks are formed on the fly, devices can leave and join the network during its lifetime, devices can be mobile within the network, the network as a whole may be mobile and the network can be deformed on the fly. Devices in mobile ad hoc networks should be able to detect the presence of other devices and perform the necessary set-up to facilitate communications and the sharing of data and services. Due to the lack of continuous communications among mobile nodes and possible errors in the estimation of nodal contact probability, convergence and stability become major challenges in distributed clustering in DTMN. Extensive simulations are carried out to evaluate the end to end delay and overhead of the proposed Ad hoc On demand Distance Vector (AODV) routing protocol.

Key words: Clustering, delay-tolerant networks, routing, AODV, mobile interchangeable

INTRODUCTION

The a natural consequence of intermittent connectivity among mobile nodes, especially under low nodal density and/or short radio transmission range, the Delay-Tolerant Network (DTN) technology has been introduced to mobile wireless communications such as ZebraNet, Shared Wireless Info-Station (SWIM) Delay/Fault-Tolerant Mobile Sensor Network (DFT-MSN) and mobile internet and peer to peer mobile ad hoc networks. DTN is fundamentally an opportunistic communication system where communication links only exist temporarily, rendering it impossible to establish end to end connections for data delivery. In such networks, routing is largely based on nodal contact probabilities (or more sophisticated parameters based on nodal contact probabilities). The key design issue is how to efficiently maintain, update and utilize such probabilities. Most DTN protocols are at where every node plays a similar role in routing. The flat architecture is simple and effective in small networks but not scalable to large size DTNs (Fall, 2003).

Meanwhile, clustering has long been considered as an effective approach to reduce network overhead and improve scalability. Various clustering algorithms have been investigated in the context of mobile ad hoc networks. However, none of them can be applied directly to DTN because they are designed for well-connected networks and require timely information sharing among nodes. A recent research proposes a DTN Hierarchical Routing (DHR) protocol to improve routing scalability. DHR is based on a Deterministic Mobility Model where all nodes move according to strict, repetitive patterns which are known by the routing and clustering algorithms. It cannot be generalized to such networks with unknown mobility as DTN-based peer to peer mobile ad hoc networks.

In this research, researchers investigate distributed clustering and cluster-based routing protocols and AODV routing protocol for Delay-Tolerant Mobile Networks (DTMNs).

DISTRIBUTED CLUSTERING

In this study, researchers introduce clustering algorithm for DTMN which undergoes the following steps. First, each node learns direct contact probabilities to other nodes. It is not necessary that a node stores contact information of all other nodes in network. Second, a node decides to join or leave a cluster based on its contact probabilities to other members of that cluster. Since, the objective is to group all nodes with high pair

Corresponding Author: S. Kannan, SNS College of Technology, Coimbatore, Tamilnadu, India

wise contact probabilities together, a node joins a cluster only if its pair wise contact probabilities to all existing members are greater than a threshold γ. A node leaves the current cluster if its contact probabilities to some cluster members drop below γ. Finally, once clusters are formed, gateway nodes are identified for inter-cluster communications. Two clusters communicate to each other mostly via gateways (Fall, 2003; Burleigh *et al*., 2003).

Problem statement: In an intermittent environment, end to end connections do not always exist. This uniqueness leads to several main challenges for clustering and routing as elaborated.

Online estimation of contact probabilities: Pair wise contact probability has been widely used as a routing parameter in opportunistic networks. Researchers also make use of it in the algorithm. However, one of the major problems in DTN is how to obtain this parameter distributively.

A naive approach is to keep the entire meeting history. This approach while providing robustness is costly in storage and lacks agility to adapt to changes in mobility pattern. Therefore, we adopt a simple and effective approach, named Exponentially Weighted Moving Average (EWMA). More specifically, Node i maintains a list of contact probabilities ξ_{ij} for every other node j which it has met before. ξ_{ij} is updated in every time slot according to the following rule:

$$\xi_{ij} = \begin{cases} (1-\alpha)[\xi_{ij}]+\alpha, & \text{i meets j} \\ (1-\alpha)[\xi_{ij}], & \text{otherwise} \end{cases} \quad (1)$$

where is a constant parameter between 0 and 1 and $[\xi_{ij}]$ is the old contact probability. Clearly this is a dynamic process and thus ξ_{ij} doesnt necessarily equal to the actual contact probability p_{ij}. However, the studies show EWMA effectively yields ξ_{ij} whose mean converges to p_{ij} which is important to ensure stable clusters. Formally, we establish the following theorem (Burleigh *et al*., 2003; Wang and Wu, 2006).

Theorem 1: If Nodes i and j have a probability of p_{ij} to meet in each time slot, EWMA yields ξ_{ij} whose mean converges to p_{ij}.

Proof: Consider a sequence of time slots and let ξ_{ij} (t) denote ξ_{ij} in time slot t. Clearly, the mean of ξ_{ij} (1) is:

$$E(\xi_{ij}(1)) = (1-\alpha)\xi_{ij}(0) + p_{ij}\alpha$$

Similarly, we have:

$$E(\xi_{ij}(2)) = (1-\alpha)^2\xi_{ij}(0) + p_{ij}\alpha[1+(1-\alpha)]$$

And:

$$E(\xi_{ij}(t) = (1-\alpha)^t \xi_{ij}(0) + p_{ij}\alpha[1+(1-\alpha)+...+(1-\alpha)^{t-1}]$$

Let $t\to\infty$, we arrive at:

$$\lim_{t\to\infty} E(\xi_{ij}(t)) = \alpha p_{ij}\frac{1}{\alpha} = p_{ij}$$

which depends on neither the parameter α nor the initial value ξ_{ij} (0). Although, EWMA is proven to eventually arrive at the average which is equal to the nodal contact probability and the convergence is independent of α, it is in fact a random process that fluctuates around its mean value. Such fluctuation may lead to an estimation error of the nodal contact probability at a particular time. As a result, there is a trade-off in the selection of α. A smaller α results in less errors but at the same time, longer delay to reach the steady state. In general, α should be chosen according to applications and application-specific requirements.

Fractional clusters: Due to possible errors in the estimation of contact probabilities and unpredictable sequence of the meetings among mobile nodes, many unexpected small size clusters may be formed. To deal with this problem, we employ a merging process that allows a node to join a better cluster where the node has a higher stability. The merging process is effective to avoid fractional clusters (Wang and Wu, 2007).

Inconsistent cluster membership and gateway selection: The problem of inconsistency may appear in both cluster membership and gateway selection. For example, Node j leaves its current Cluster C_q and joins the new Cluster C_p. Given the network with low connectivity, other members of C_q are not timely informed of this change and thus falsely assume that Node j is still a cluster member. The inconsistency problem exists also in gateway selection for a similar reason. For instance, two nodes in the same cluster may have two different gateways to another cluster. Or a node may lose its gateway to an adjacent cluster because the gateway node has left. We deal with the inconsistency problems by employing a synchronization mechanism where nodes exchange and keep only the most up to date information (Wang and Wu, 2007).

Cluster member with low contact probability: A node with a very low nodal contact probability may still appear

in the member list of another node. The main reason is that a mobile node may change its mobility pattern in real-life applications. For example, a student may have his/her regular mobility pattern in a semester (visiting certain class rooms, libraries, dormitories and cafeterias, etc.). When entering a new semester, his/her mobility may change due to the new/rescheduled classes and after school activities. Similar scenarios will happen at holidays, summer/winter breaks or when the student changes his/her major. When mobility pattern changes but the member list is yet updated, the problem stated above may happen. A possible solution for this problem is to use timeout for membership binding. We will discuss this problem in this study (Lindgren *et al*., 2004).

Clustering meta-information: Without loss of generality, a node, e.g., Node i, maintains its ID (i.e. i), its cluster ID (denoted by Ω (i)), a cluster table and a gateway table as its local information.

Node i maintains its gateway information in the gateway table with four fields: Cluster ID, Gateway, Contact Probability and Time Stamp. The Cluster ID field contains the list of clusters known by i. For each cluster, e.g., Cluster c, the Gateway field includes the ID of the gateway (denoted by G_i^c) while the contact probability field (C_c^i) indicates the highest contact probability between the gateway G_i^c and any node in Cluster c. Similarly, G_i^c and C_c^i are updated according to the best knowledge of Node i and thus not necessary to be network-wide correct/accurate. Finally, the Time Stamp field contains the most recent time when the entry is updated.

Distributed clustering algorithm: The key part of the algorithm lies on the meeting event between any pair of nodes. A node then decides its actions subsequently. Specifically, a node will join a new cluster if it is qualified to be a member. Similarly, a node leaves its current cluster if it joins a new cluster or it is no longer qualified to be in the current cluster. When two member nodes meet, they trigger the synchronization process to update their information. To this end, researchers de ne three main functions, namely join, leave and sync for the algorithm. During initialization, Node i creates a cluster that consists of itself only and two empty tables. Its cluster ID is set to be its node ID appended with a sequence number, Ω (i) = i:Seq. Each node maintains its own sequence number which increases by one whenever the node creates a new cluster to avoid duplication. The algorithm is event-driven. Hereafter, node i waits for three possible events, i.e., Slot-Timeout, Meet A Node and Gateway Outdate (Wang and Wu, 2007; Lindgren *et al*., 2004).

Slot Timeout Event (Update contact probability): A Slot Timeout event is generated by the end of every time slot, triggering the process of updating the contact probabilities by using the EWMA.

Once the contact probabilities are updated, the GatewayUp-date() procedure is invoked to update the gateway table. As discussed earlier, the gateway table maintains a list of gateways to each cluster. Since, node has updated its contact probabilities to all nodes, it may potentially choose better gateways. During this procedure, the following three cases are considered:

Case 1: Consider an entry in the gateway table, e.g., for Cluster c. If Node i is not the gateway to Cluster c ($G_i^c \neq i$), it looks up the cluster table to identify Node k, to which it has the highest contact probability, among all nodes in Cluster c, i.e., $\xi_{ik} \geq \xi_{ik'}$, ¥k, k' $\in C_c^i$. If $\xi_{ik} > C_c^i$ (the current gateway probability), the entry is updated by setting $G_i^c = k$, $C_c^i = \xi_{ik}$ and T_i^c to be the current clock time.

Case 2: Still consider an entry for Cluster c in the gateway table of Node i. If Node i itself is the gateway ($G_i^c = i$). While Node i still identifies Node k with the highest contact probability as discussed above. If $\xi_{ik} \geq \hat{\gamma}$ (the gateway threshold), the entry is updated by setting $G_i^c = k$, $\zeta_i^c = \xi_{ik}$ and T_i^c to be the current clock time; otherwise G_i^c = N/A, $\zeta_i^c = 0$ and T_i^c = N/A (or simply removes the entry). This is particularly important to keep the gateway information up to date.

An example is shown in Fig. 1 where Nodes i and k were the gateways between their clusters before Node k joins another cluster without notifying the change to Node i. If they meet again, Node i learns the change from Node k and thus updates its tables; otherwise if they do not meet each other for a long time, ξ_{ik} becomes lower than $\hat{\gamma}$. In either case, Node i's gateway table should be updated correctly by using the above procedure.

Case 3: Node also checks the cluster table for possible new clusters not included in its current gateway table. If a new cluster is found, it will be added into the gateway table.

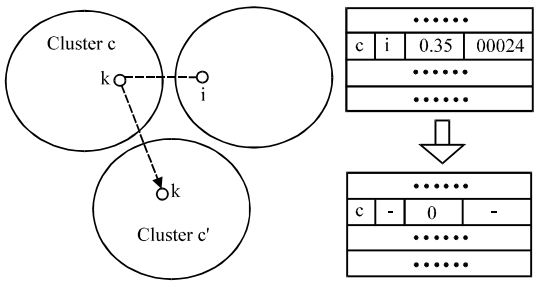

Fig. 1: Update of the gateway table

Meet A Node Event (Update cluster information): The Meet A Node Event is generated upon receiving the Hello message (exchanged between two meeting nodes, i and j). A series of actions will be taken at both sides as elaborated.

If Nodes i and j are in the same cluster, $\Omega(i) = \Omega(j)$, the membership check function is invoked to verify if they are still qualified to stay in the same cluster. Specially if $\xi_{ik} \geq \hat{\gamma}$ they continue to stay in the sane cluster and perform the synchronization of their cluster information.

The Sync() procedure includes two steps for synchronizing cluster members and gateways, respectively.

Synchronization of cluster members: Note that although, $\Omega(i) = \Omega(j)$ $C^i_{\Omega(i)}$ and $C^j_{\Omega(j)}$ (the lists of cluster members at Nodes " and $, respectively) are not necessarily identical because the clustering is distributed and thus Nodes i and j may maintain different cluster information before the clustering procedure is converged. The basic idea of synchronization is to update the membership based on the latest information of Nodes i and j. More specifically, Node i sends to Node j a set of its current cluster members along with the time stamps, i.e., $\Psi = \{(k, T_i^k) | k \in C^i_{\Omega(i)}\}$. Upon receiving Ψ, Node j divides it into two subsets based on time stamps $\Psi_1 = \{(k, T_i^k) | k \in C^i_{\Omega(i)}, k \notin C^j_{\Omega(j)}, T_i^k > T_j^k\}$ and $\Psi_2 = \{(k, T_i^k) | k \in C^i_{\Omega(i)}, k \notin C^j_{\Omega(j)}, T_i^k < T_j^k\}$. The former is the set that Node i has newer information (as $T_i^k > T_j^k$) while Node j's information on the latter is newer. As a result, Node $ updates its cluster table according to Ψ by setting $\Omega_i^k = \Omega(i) = \Omega(j), \forall k \in \Psi_1$. Meanwhile, it sends $\{(k, \Omega_j^k) | k \in \Psi_2\}$ to Node i which in turn updates its cluster table by setting $\Omega_i^k = \Omega_j^k \ \forall \ k \in \Psi_2$. Node j in turn sends its current cluster members to Node i for a similar process.

Synchronization of gateways: Nodes i and j may have different gateways to the same cluster (s) in their gateway tables. Thus, synchronization is needed to keep the better. one with higher contact probability. For each of such clusters, the node whichever has lower contact probability gives up and updates its gateway table. For example, consider Cluster c and assume $\zeta_i^c < \zeta_j^c$. Then, Node i updates its gateway table by setting $G_i^c = G_j^c$ and $\zeta_i^c < \zeta_j^c$. However, the Time Stamp, T_i^c is not updated unless Node j is the gateway itself.

If Nodes i and j do not pass the membership check, one of them must leave the cluster. Two issues are involved in the Leave() procedure. First, we identify the leaving node based on its stability, defined as its minimum contact probability with its cluster members, i.e., $\min\{\xi_{ik} | k \in C_{\Omega(i)}\}$ for Node i. The node with lower stability leaves. It indicates the likelihood that the node will be excluded from the cluster due to its low contact probability. If there is a tie, the node with higher ID is chosen. Second, the leaving node creates a new cluster that consists of itself only. It keeps the current cluster table because all information in the table is still valid and resets the gateway table to be empty. Then, it sends its new cluster ID to the other node to update the cluster table accordingly.

If $\Omega(i) \neq \Omega(j)$, Nodes i and j consider whether or not to join each other.'s cluster. The Join() procedure is employed for a node to join a better cluster or to merge two separate clusters. It includes three steps. First, they exchange their member list ($C^i_{\Omega(i)}$ and $C^j_{\Omega(j)}$)and perform membership check. For example, Node i calculates $\xi_{ik}, \ \forall \ k \in C^j_{\Omega(j)}$. If any ξ_{ik} is $<\gamma$, the membership check procedure returns false. If none of them pass the membership check, no further action is taken. If a node passes, it goes ahead to perform Join() function. If both nodes pass, the one with lower stability joins the other cluster. Second, the selected node updates its Cluster ID and cluster table. For example, assume Node " joins Node $.'s cluster. It sets $\Omega(i) = \Omega(j)$ and $\Omega_i^k = \Omega(j), \forall k \in C^j_{\Omega(j)}$. Third, it copies the gateway table from Node j.

Gateway Outdate Event (Reset gateway): When the Time Stamp of any entry in the gateway table is older than a threshold, Δ, a Gateway-Outdate event is generated for that entry. For example if $t - T_i^c > \Delta$ where t is the current clock time, the gateway G_i^c is likely lost. Thus, the gateway table entry for Cluster c is reset by letting G_i^c = N/A, $\zeta_i^c = 0$ and T_i^c = N/A.

CLUSTER-BASED ROUTING PROTOCOL

Once the clustering procedure is finished, each node in the network is associated with a cluster. For any two clusters whose members have high enough contact probability ($\geq \hat{\gamma}$) a pair of gateway nodes are identified to bridge them. In this study, researchers discuss how to route data messages efficiently in DTN by utilizing the clusters and gateways.

Consider Node i which intends to send a data message to Node j. Node i looks up its cluster table to find the cluster ID of Node j, i.e., Ω_i^j. According to Ω_i^j, three scenarios are considered: intra-cluster routing, one-hop inter-cluster routing and multi-hop inter-cluster routing (Lindgren *et al*., 2004).

Intra-cluster routing: If $\Omega_i^j = \Omega(i)$, Nodes i and j are in the same cluster (Fig. 2a). Since all nodes in a cluster have high contact probability, direct transmission is employed

here. In other words, Node i transmits the data message only when it meets Node j. No relay node is involved in such intra-cluster routing.

One-hop inter-cluster routing: If $\Omega_i^j \neq \Omega$ (i), Node i looks up its gateway table. If an entry for Ω_i^j is found, there exists a gateway, i.e., $G_i^{\Omega^j}$ to Node j's cluster. In this scenario, Node i sends the data message to its gateway. Upon receiving the data message, the gateway looks up its gateway table to find Node j's cluster ID. Whenever, it meets any node, e.g., Node k in Node j's cluster, it forwards the message to Node k which in turn delivers the data message to Node j through intra-cluster routing as discussed above. Since, Node 6#! " ! is the gateway, it has high probability to meet at least one node in Node j's cluster. Note that Node k in Fig. 2b is not necessary to be the gateway node.

Multi-hop inter-cluster routing: If $\Omega_i^j \neq \Omega$ (i) and Node i does not find Ω_i^j in its gateway table, the approaches discussed so far will fail to deliver the data message because the destination (Node j) is not in any cluster that is reachable by Node i's gateways. As a result, the data transmission from Node i to Node j needs to be devised for multi-cluster routing. Given the low connectivity in delay-tolerant mobile networks, on-demand routing protocols do not work effectively here because the flooding-based on-demand route discovery leads to extremely high packet dropping probability as shown in. On the other hand, any table-driven routing algorithms may be employed for multi-hop inter-cluster routing. For simplicity, a link-state-like routing scheme is used as an example in the following discussions. In the protocol, every gateway node builds and distributes a Cluster Connectivity Packet (CCP) to other gateways in the network. The CCP of Gateway i comprises its cluster ID and a list of clusters to which it serves as the gateway and the corresponding contact probabilities, $\{(c, \zeta_i^c) | \forall\ G_i\ ^c = i\}$. Such information can be readily obtained from the gateway table. Examples of CCP are shown in Fig. 2c. Note that the actual implementation of CCP also includes a sequence number to eliminate outdated information (Lindgren *et al*., 2004).

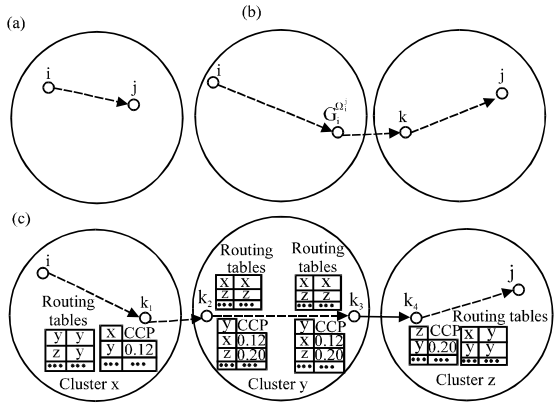

Fig. 2: Cluster-based routing; a) Intra-cluster routing; b) One-hop intra-cluster routing; c) Multi-hop intra-cluster routing

AD-HOC ON-DEMAND DISTANCE VECTOR (AODV)

Ad hoc On-demand Distance Vector (AODV) which is used to provide secure and reliable data transmission over the MANETs. AODV discovers a route through network wide broadcasting. The source host starts a route discovery by broadcasting a route request to its neighbors. In the route request, there is a requested destination sequence number which is 1 greater than the destination sequence number currently known to the source. This number prevents old routing information being used as reply to the request which is the essential reason for the routing loop problem in the traditional distance vector algorithm.

When a node wants to send a packet to some destination node and does not have a valid route in its routing table for that destination, it initiates a route discovery process. Source node broadcasts a Route Request (RREQ) packet to its neighbours which then forwards the request to their neighbours and so on. Nodes generate a route request with destination address, sequence number and broadcast ID and sent it to his neighbour nodes. Each node receiving the route request sends a route back (Forward Path) to the node as shown in the Fig. 3. When the RREQ is received by a node that is either the destination node or an intermediate node with a fresh enough route to the destination, it replies by unicasting the Route Reply (RREP) towards the source node. As the RREP is routed back along the reverse path intermediate nodes along this path set up forward path entries to the destination in its route table and when the RREP reaches the source node, a route from source to the destination established Fig. 4 shows the path of the RREP from the destination node to the source node.

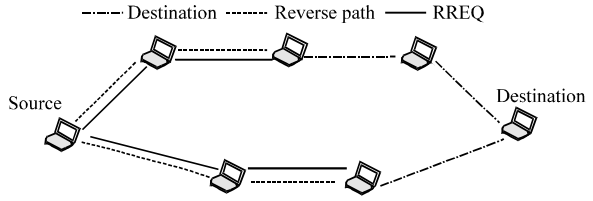

Fig. 3: Route requests in AODV

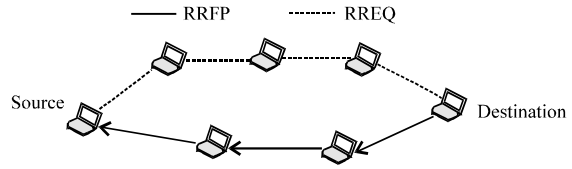

Fig. 4: RREP in AODV

CONCLUSION

Researchers have investigated clustering and cluster-based routing in DTMN. Due to the lack of continuous communications among mobile nodes and possible errors in the estimated nodal contact probability, convergence and stability become major challenges in distributed clustering in DTMN. To this end, an Exponentially Weighted Moving Average (EWMA) scheme has been employed for on-line updating the contact probabilities with its mean proven to converge to the true contact probability. Based on contact probabilities, a set of functions including Sync(), Leave() and Join() has been devised for cluster formation and gateway selection. Finally, the gateway nodes exchange network information and perform routing. There is always a delay occurring in Delay Tolerant Mobile Networks. Using AODV which is an efficient algorithm for ad hoc networks we minimize the delay in DTMN and provide a scalable routing.

REFERENCES

Burleigh, S., A. Hooke, L. Torgerson, K. Fall and V. Cerf *et al*., 2003. Delay-tolerant networking: An approach to interplanetary Internet. IEEE Commun. Mag., 41: 128-136.

Fall, K., 2003. A delay-tolerant network architecture for challenged internets. Proceedings of the Conference on Applications, Technologies, Architectures and Protocols for Computer Communications, August 25-29, 2003, Karlsruhe, Germany, pp: 27-34.

Lindgren, A., A. Doria and O. Schelen, 2004. Probabilistic Routing in Intermittently Connected Networks. In: Service Assurance with Partial and Intermittent Resources, Dini, P. *et al.* (Eds.). LNCS., 3126, Springer-Verlag, Berlin, Heidelberg, ISBN: 978-3-540-22567-6, pp: 239-254.

Wang, Y. and H. Wu, 2006. DFT-MSN: The delay/fault-tolerant mobile sensor network for pervasive information gathering. Proceedings of 25th IEEE International Conference on Computer Communications, April 23-29, 2006, Barcelona, Spain, pp: 1-12.

Wang, Y. and H. Wu, 2007. Delay/fault-tolerant mobile sensor network (DFT-MSN): A new paradigm for pervasive information gathering. IEEE Trans. Mobile Comput., 6: 1021-1034.