Featured Concurrency of Mobile ad hoc Computing

Phan Cong Vinh
NTT University, Vietnam
pcvinh@ntt.edu.vn

ABSTRACT
Mobile ad hoc computing (MAC) is a form of concurrent computing by which computational processes are executed in parallel by assigning each computational process to one of mobile devices on a mobile ad hoc network (MANET). The overall goal of MAC is to support such MANETs capable of management and high performance. Meeting this grand challenge of MAC requires that MANET processes concurrency not tackled before is necessarily featured. To this end, this paper presents a firm formal development for featuring concurrency of MANET processes.

Categories and Subject Descriptors
D.4.1 [Process Management]: Concurrency

General Terms
Theory

Keywords
Concurrency, Device-to-Device (D2D), MANET, Mobile ad hoc computing (MAC), Self-configuration

1. INTRODUCTION
Mobile applications are currently booming. As mobile services and communications infrastructure become richer, as mobile devices become more powerful, and as consumer expectations evolve, we are faced with an array of challenges that affect how to manage MANET behaviors efficiently. Mobile ad hoc computing (MAC) is seen as an essential computing paradigm to keep such MANETs capable of management and high performance [2]. In fact, MAC is only possible when mobile devices autonomously interact and coordinate with each other to maintain properly the required computations [1]. The essence of MAC is to enable the mobile devices to execute concurrently assigned computational processes and deliver shared resources while interacting and coordinating with each other. Hence, for MANETs, one of major challenges is how to support processes concurrency in the face of changing location and context frequently. With this aim, we develop a firm formal approach in which the notions of concurrency of MANET processes and self-configuration are featured and specified in categorical structures. The major contribution of the paper is to propose some applied categorical structures for featuring concurrency of MANET processes.

The rest of this paper is organized as follows: In section 2, we present clearly and exactly concurrency of MANET processes and category of MANET processes. Section 3 concentrates on Device-to-Device (D2D) networks on MANET together with categorical aspects of self-configuration. Finally, a short summary is given in section 4.

2. CONCURRENCY OF PROCESSES

MAC is a type of concurrent computing in which computations are designed as collections of interacting computational processes that may be executed in parallel. Concurrent computations can be executed in parallel by assigning each computational process to one of mobile devices distributed across an MANET. The main challenges in designing MAC are ensuring the correct sequencing of the interactions or communications between different computational processes and coordinating access to resources that are shared among processes in heterogeneous environments, decentralization, networking links of varying latencies, unpredictable failures and dynamicity of MANETs.

DEFINITION 1. A concurrency of processes on an MANET is a binary operation (denoted as $\parallel$) between two processes on the MANET which groups them together as being parallel in the mobile environments. Hence, let $a$ and $b$ be arbitrary processes on the MANET, then $a \parallel b$ denotes that $a$ and $b$ are executed concurrently.

It follows that the concurrent composition of a series of processes $a_1, a_2, \ldots, a_n$ $a_1 \parallel (a_2 \parallel (\ldots \parallel a_n) \ldots)$, is simply written as $a_1 \parallel a_2 \parallel \ldots \parallel a_n$ or $\big|_{1 \leq i \leq n} a_i$. Sometimes, the notation of $\parallel$ is used to denote a special process skip that has no effect on any state of MANET, and terminates immediately.

The algebraic laws [3] governing the behavior of $a \parallel b$ are exceptionally simple and regular. The following are three laws related to our development in this section.

- Law on symmetry: This law expresses the logical symmetry between two processes.
\begin{equation}
\begin{align*}
a | b = b | a
\end{align*}
\end{equation}

- **Law on associativity**: This law shows that when three processes are assembled, it does not matter in which order they are put together

\begin{equation}
(a | b) | c = a | (b | c)
\end{equation}

- **Law on identity**: Composition with `skip` makes no difference

\begin{equation}
a | \text{skip} = a
\end{equation}

For every process `a` and `b` on an MANET, if `a` and `b` are executed concurrently then we define a labeled arrow from `a` to `b` as `a \xrightarrow{a|b} b` in which its label is `a | b`. Category of MANET processes is founded upon the abstraction of the labeled arrow called *morphism*. Here, the labeled arrow \( a \xrightarrow{a|b} b \) is the morphism between `a` and `b`. We usually use two following diagrams to specify this morphism.

\begin{equation}
a \xrightarrow{a|b} b \quad \text{or} \quad a \mid b : a \longrightarrow b
\end{equation}

where source and target of the morphism `a \mid b` are the processes `a` and `b`, respectively. Such directional structures occur widely in representation of this paper.

By describing structures in terms of the existence and characteristics of morphisms \( a \mid b \), categorical structures of MANET processes achieve their wide applicability. The usual method of mathematical description is by reference to the internal structure of MANET processes. The applicability of this description is then limited to MANET processes supporting such structure. Categorical descriptions make no assumption about the internal structure of MANET processes, but they purely ensure that whatever structure of MANET processes is preserved by the morphisms \( a \mid b \). In this sense, categorical representations are data independent descriptions. Thus the same description may apply to whatever can be seen as MANET processes in a category.

A category of MANET processes, which is a fundamental and abstract way to describe processes of an MANET and their relationships, is composed of a set of MANET processes (also called objects) together with a set of morphisms (sometimes called arrows) \( a \mid b \) between MANET processes. Morphisms \( a \mid b \) are to be composable, that is, if `a \mid b : a \longrightarrow b` and `b \mid c : b \longrightarrow c`, then there is a concurrent composition `a \mid b \mid c` such that the laws on associativity in (2) and on identity in (3) are satisfied.

Category of MANET processes is thus a directed graph with the concurrent composition and identity structure. This leads to the following formal definition of the category, named \( \text{maP} \), of MANET processes. We name this category \( \text{maP} \) to refer to “mobile ad hoc Processes”.

**Category \( \text{maP} \)** is a graph \((\text{Processes}, \text{ConCom}, s, t)\) consisting of Processes which is the set of processes (considered as nodes). ConCom is the set of concurrent compositions (considered as edges) and \( s, t: \text{ConCom} \longrightarrow \text{Processes} \) are two maps called source (or domain) and target (or codomain), respectively, such that the following axioms (also called coherence statements) hold:

- (Associativity) If `a \mid b \mid b \mid c` then `(a \mid b) \mid c = a \mid (b \mid c)`. For notational convenience, this can be written as

\begin{equation}
\begin{align*}
a \mid b \quad \text{and} \quad b \mid c \\
(a \mid b) \mid c = a \mid (b \mid c)
\end{align*}
\end{equation}

- (Identity) For every process `a`, there exists a morphism `a \mid \text{skip}` called the identity morphism for `a`, such that for every morphism `a \mid b`, we have

\begin{equation}
a \mid \text{skip} \mid b = a \mid b = a \mid b \mid \text{skip}
\end{equation}

We write \( a \mid b : a \longrightarrow b \) when `a \mid b` is in \( \text{ConCom} \) and \( s(a \mid b) = a \) and \( t(a \mid b) = b \).

**Property 1.** The category \( \text{maP} \) is just a device-to-device (D2D) network on MANET

**Property 2.** The category \( \text{maP} \) is a complete graph

### 3. DEVICE-TO-DEVICE NETWORKS

Categorical characteristics are often expressed in terms of commutative diagrams and justifications take the form of diagram chasing. Informally, a diagram is a picture of some processes and morphisms in the category \( \text{maP} \). Formally, a diagram is a graph whose nodes are labeled with processes of \( \text{maP} \) and whose edges are labeled with morphisms of \( \text{maP} \) in such a way that source and target processes of an edge are labeled with source and target processes of the labeling morphism.

Laws on symmetry in (1), on associativity in (2) and on identity in (3) are respectively represented by the following commutative diagrams:

\begin{equation}
\begin{align*}
\text{Diagram 1:}
\end{align*}
\end{equation}

\begin{equation}
\begin{align*}
\text{Diagram 2:}
\end{align*}
\end{equation}

A path in a diagram is a non-empty sequence of edges and their labeling morphisms such that the target process of each edge is the source process of the next edge in the sequence. For example, the central diagram in (7) contains the path `a \xrightarrow{a|b} b \xrightarrow{b|c} c`.

Each path determines a morphism by composing the morphisms along it. A diagram is said to commute if, for every pair of processes `x, y`, every path from `x` to `y` determines the same morphism through composition.

The category \( \text{maP} \), which consists of the set Processes of processes together with morphisms \( a \mid b \) in the set \( \text{ConCom} \), generates D2D structure on MANET. The D2D structure is dynamic in nature because processes can be dynamically added to or dropped from MANET. For such every change, self-configuration [5] for the D2D structure on MANET occurs.

### 3.1 Self-configuration of D2D networks

Let \( \text{OP} = \{\text{add}, \text{drop}\} \) be the set of operations making a D2D structure on MANET change, in which add and drop are defined as follows:

\begin{equation}
\text{add} : \text{ConCom} \times \text{Processes} \longrightarrow \text{ConCom}
\end{equation}
obeying the following axioms: For all \( i \in \mathbb{N}_0, \)
\[
\text{\text{add}}(\emptyset, a_i, b) = \left\{ \begin{array}{ll}
|1 \leq i \leq (n-1)| a_i & b \\
|a_0| b = \text{skip} & b = b \\
\text{when } i = 0
\end{array} \right. \tag{9}
\]
drop is a binary operation
\[
\text{drop} : \text{ConCom} \times \text{Processes} \rightarrow \text{ConCom} \tag{10}
\]
obeying the following axioms: For all \( i \in \mathbb{N}_0, \)
\[
\text{drop}(\emptyset, a_i, b) = \left\{ \begin{array}{ll}
|1 \leq i \leq (n-1)| a_i & b \\
|a_1| \text{when there exists } a_i = b & a_i, \text{ for all } a_i \neq b
\end{array} \right. \tag{11}
\]
It follows that \( \text{drop}(\emptyset, b) = \emptyset = \text{skip}. \) A process of self-configurations is completely defined when operations \text{add} and \text{drop} are executed on an MANET as illustrated in the following diagram:
\[
\begin{array}{cccccc}
\text{add} & \text{drop} & \text{add} & \text{drop} & \text{add} & \text{drop} \\
(\emptyset) & (a_1) & (a_2) & (\emptyset) & (a_3) & (a_1)
\end{array}
\tag{12}
\]

In the context of \text{maP}, self-configurations are known as homomorphisms from a \text{maP} to another \text{maP} to preserve the D2D structure. In other words, self-configuration is a map from a ConCom to another ConCom of the same type that preserves all the concurrent structures.

### 3.2 Category of D2D networks

Further to the category \text{maP}, a category whose objects are categories \text{maP} and whose morphisms are self-configurations is called the category \text{Cat(maP)} of D2D networks.

The category \text{Cat(maP)} is constructed as follows:
- **Objects as categories \text{maP}**: Let \text{ObjCat} be the set of all categories \text{maP}. That is,
\[
\text{ObjCat} = \{\text{maP} | \text{maP} \text{ is a category of MANET processes} \} \tag{13}
\]

- **Morphisms as self-configurations**: Associated with each pair of categories \text{maP} and \text{maP'} in \text{ObjCat}, self-configuration \( h : \text{maP} \rightarrow \text{maP'} \) to map every D2D structure to another is a self-configuration from \text{maP} to \text{maP'} such that for all concurrent compositions \( a \mid b \) in \text{ConCom} it holds that \( h(a \mid b) = h(a) | h(b) \).

For each pair of self-configurations \( h : \text{maP} \rightarrow \text{maP'} \) and \( k : \text{maP} \rightarrow \text{maP''} \), there is an associated self-configuration \( h \cdot k : \text{maP} \rightarrow \text{maP''} \), the composition of \( h \) with \( k \) (and read as “\( h \) before \( k \)”), such that for all concurrent compositions \( a \mid b \) in ConCom it holds that \( k(h(a) | h(b)) = k(h(a)) \mid (k(h(b)) \).

Associated with each \text{maP} in \text{ObjCat}, then self-configuration \text{id}_{\text{maP}} : \text{maP} \rightarrow \text{maP} to map every D2D structure to itself is an identity self-configuration from \text{maP} to \text{maP} such that for all concurrent compositions \( a \mid b \) in ConCom it holds that \( \text{id}_{\text{maP}}(a \mid b) = \text{id}_{\text{maP}}(a) \mid \text{id}_{\text{maP}}(b) = a \mid b \).

As a result, for every D2D structure and the self-configurations \( h : \text{maP} \rightarrow \text{maP'}, k : \text{maP} \rightarrow \text{maP''} \) and \( g : \text{maP''} \rightarrow \text{maP''} \), the following equations must hold

**Associativity**: \( (h \cdot k) \cdot g = h \cdot (k \cdot g) \)

**Identity**: \( \text{id}_{\text{maP}} \cdot h = h = h \cdot \text{id}_{\text{maP}} \)

These two equations amount to two following commutative diagrams, respectively.

\[
\begin{array}{cccccc}
\text{maP} & \text{maP'} & \text{maP''} & \text{maP''} \\
\downarrow{h} & \downarrow{h} & \downarrow{g} & \downarrow{g} \\
\text{maP} & \text{maP'} & \text{maP''} & \text{maP''}
\end{array}
\tag{14}
\]

and

\[
\begin{array}{ccc}
\text{id}_{\text{maP}} & \text{id}_{\text{maP}} \\
\downarrow{h} & \downarrow{h} \\
\text{maP} & \text{maP'}
\end{array}
\tag{15}
\]

These are all the basic ingredients we need for the category \text{Cat(maP)} of D2D networks defined.

**Property 3.** Every self-configuration \( h : \text{maP} \rightarrow \text{maP'} \) and \( g : \text{maP} \rightarrow \text{maP''} \) in \text{Cat(maP)}, there exist unique self-configurations \( x : \text{maP} \rightarrow \text{maP} \) and \( y : \text{maP} \rightarrow \text{maP''} \) in \text{Cat(maP)} such that the following equation holds

\[
x \cdot h \cdot y : \text{maP} \rightarrow \text{maP''} = g : \text{maP} \rightarrow \text{maP''}
\tag{16}
\]

It follows that the self-configurations \text{id}_{\text{maP}} : \text{maP} \rightarrow \text{maP} and \text{id}_{\text{maP'}} : \text{maP'} \rightarrow \text{maP'} are identity self-configurations, then the equation

\[
\text{id}_{\text{maP}} \cdot h \cdot \text{id}_{\text{maP'}} : \text{maP} \rightarrow \text{maP'} = h : \text{maP} \rightarrow \text{maP'}
\tag{17}
\]

**Property 4.** For all self-configurations \( x \) and \( y \) in \text{Cat(maP)}, if \( x \cdot h \cdot y = x \cdot h' \cdot y \), then \( h = h' \)

These properties of self-configurations in \text{Cat(maP)} direct towards a categorical structure, called extensional monoidal category as presented in the next subsection.

### 3.3 Extensional monoidal category of D2D networks

Further to the category \text{Cat(maP)} of D2D networks, we investigate the extensional monoidal structure of the category \text{Cat(maP)} of D2D networks. The operation “\( \cdot \)” defines an extensional monoidal structure on the category \text{Cat(maP)}. In fact, \text{Cat(maP)} equipped with the following multifunctor defines an extensional monoidal category. (Note that a monoidal category is named as “extensional monoidal category” when it is equipped with a multifunctor, in general, for a distinction from a normal monoidal category just with a bifunctor)

\[
: \text{Cat(maP)} \times \text{Cat(maP)} \times \text{Cat(maP)} \rightarrow \text{Cat(maP)}
\tag{18}
\]

which, called composition operation, is associative up to a natural isomorphism, and an identity self-configuration \text{id} which is both a left and right identity for the multifunctor “\( \cdot \)”, again, up to natural isomorphism. The associated natural isomorphisms are subject to some coherence conditions which ensure that all the relevant diagrams commute. We consider the facts in detail as below.
Property 5. The composition operation “•” is associative up to three natural isomorphism \( \alpha, \beta, \gamma \), called associative ones, with components:

\[
\begin{align*}
\alpha(h, k, g, d, e) : (h, k, g) \cdot d \cdot e & \cong h \cdot (k, g) \cdot d \cdot e \\
\beta(h, k, g, d, e) : h \cdot (k, g) \cdot d \cdot e & \cong h \cdot k \cdot (g, d) \cdot e \\
\gamma(h, k, g, d, e) : (h, k, g) \cdot d \cdot e & \cong h \cdot k \cdot (g, d) \cdot e
\end{align*}
\]

Sometimes, the natural isomorphisms \( \alpha, \beta, \gamma \) are also represented as

\[
\allowdisplaybreaks
\begin{align*}
\alpha(h, k, g, d, e) : (h \cdot k \cdot g) \cdot d \cdot e & \cong h \cdot (k \cdot g) \cdot d \cdot e \\
\beta(h, k, g, d, e) : h \cdot (k \cdot g) \cdot d \cdot e & \cong h \cdot k \cdot (g \cdot d) \cdot e \\
\gamma(h, k, g, d, e) : (h \cdot k \cdot g) \cdot d \cdot e & \cong h \cdot k \cdot (g \cdot d) \cdot e
\end{align*}
\]

The coherence conditions for three natural isomorphisms \( \alpha, \beta, \gamma \) are investigated in [4] for all self-configurations \( h, k, g, d, e, f \) and \( p \in \text{Cat}(\text{maP}) \).

Property 6. Every self-configuration \( h : \text{maP} \rightarrow \text{maP} \) has \( \text{id}_{\text{maP}} \) and \( \text{id}_{\text{maP}}^{-1} \) as left and right identity self-configurations, respectively. There is a natural isomorphism \( \lambda \), called identity one, with components:

\[
\lambda(h) : \text{id}_{\text{maP}} \cdot h \cdot \text{id}_{\text{maP}}^{-1} \cong h
\]

The coherence condition for the identity natural isomorphism \( \lambda \) is considered in [4] for all self-configurations \( h, k \) and \( g \) in \( \text{Cat}(\text{maP}) \).

The coherence condition states that two or more natural isomorphisms between two given multifunctors are equal based on the existence of which is given or follows from general characteristics. Such situations are ubiquitous in processes concurrency known as a categorical approach of the MANET processes concurrent computing.

3.4 Pushout of self-configuring D2D networks

We will investigate the concept of pushout as a categorical characteristic of self-configuring D2D networks. We firstly consider an illustration that if a D2D structure is \( a \mid b \) then \( a \mid b \xrightarrow{\text{add}(c)} a \mid b \mid c \) and \( a \mid b \xrightarrow{\text{drop}(b)} a \) are two typical self-configurations. This can be specified by a diagram (see (20)) consisting of two self-configurations \( \text{add}(c) : a \mid b \xrightarrow{\text{add}(c)} a \mid b \mid c \) and \( \text{drop}(b) : a \mid b \xrightarrow{\text{drop}(b)} a \) with a common domain.

\[
\begin{array}{c}
\xymatrix{ a \mid b \xrightarrow{\text{add}(c)} a \mid b \mid c \ar[dr]_{\text{drop}(b)} & \\
& a \mid b }
\end{array}
\]

Then applying \( \text{drop}(b) \) on \( a \mid b \mid c \) and \( \text{add}(c) \) on \( a \) we reach the same structure \( a \mid c \) as described in the diagram (21):

\[
\begin{array}{c}
\xymatrix{ a \mid b \xrightarrow{\text{add}(c)} a \mid b \mid c \ar[dr]_{\text{drop}(b)} & a \mid b \
& a \mid c }
\end{array}
\]

A pushout of a pair of self-configurations \( \text{add}(c) : a \mid b \xrightarrow{\text{add}(c)} a \mid b \mid c \) and \( \text{drop}(b) : a \mid b \xrightarrow{\text{drop}(b)} a \) is a D2D structure \( a \mid c \) together with a pair of self-configurations \( \text{drop}(b) : a \mid b \xrightarrow{\text{drop}(b)} a \mid c \) and \( \text{add}(c) : a \xrightarrow{\text{add}(c)} a \mid c \) such that the diagram (21) commutes.

Moreover, the pushout \( a \mid c \xrightarrow{\text{drop}(b)} a \mid b \) \( \cong a \mid c \) must be universal with respect to this diagram. That is, for any other pushout such as \( (b \mid c \xrightarrow{\text{drop}(a)} a \mid b \mid c, \text{add}(b), \text{add}(c), \text{drop}(a)) : a \xrightarrow{\text{add}(b)} b \mid c \) there exists a unique self-configuration \( \text{drop}(a) \), \( \text{add}(b) : a \mid c \xrightarrow{\text{add}(b)} b \mid c \) making the following diagram commutes:

\[
\begin{array}{c}
\xymatrix{ a \mid b \xrightarrow{\text{add}(c)} a \mid b \mid c \ar[rr]_{\text{add}(b)} & & a \mid b \mid c \ar[rr]_{\text{add}(b)} & & a \mid b \mid c \ar[rr]_{\text{add}(b)} & & a \mid b \mid c \ar[rr]_{\text{add}(b)} & & a \mid b \mid c \ar[rr]_{\text{add}(b)} & & a \mid b \mid c }
\end{array}
\]

As with all universal constructions, this pushout is unique up to a bijective self-configuration, i.e. isomorphism.

4. CONCLUSIONS

This paper has featured concurrency of MANET processes based on categorical structures from which its useful properties emerge. We have started with investigating the concurrent composition of processes \( |_{i \in \mathcal{A}} a_i \) on MANETs to construct every D2D structure as a category \( \text{maP} \) of MANET processes together with the notion of diagram chasing, then the extensional monoidal category \( \text{Cat}(\text{maP}) \) of such D2D structures together with categorical aspects of self-configuration has been developed in detail.

5. REFERENCES


