Filtering Algorithm for Agent-Based Incident Communication Support in Mobile Human Surveillance[1]

Duco N. Ferro[a] Catholijn M. Jonker[b]

[a] *Almende B.V., Westerstraat 50, 3016 DJ Rotterdam*
[b] *Delft University of Technology, Mekelweg 4, 2628 CD Delft*

This paper presents an ontology and a filtering algorithm used in an agent-based system to support communication in case of incidents in mobile human surveillance (MHS) domains. In those domains patrols are planned in advance but may be disrupted by unforeseen events requiring immediate attention. Managing those incidents is complicated by a number of factors. The knowledge, information or support required for dealing with incidents is distributed across the organizations involved. The availability of resources changes over time, and so does the context. In addition, due to organizational and legal requirements, incidents need to be resolved within a certain time limit. Due to these problems, individuals need to initiate communication on the basis of incomplete and uncertain information.

To improve the efficiency of information provision in mobile human surveillance networks, we developed an agent-based communication management architecture that is sustained by real-time self-organization for our case study application domain: MHS security domain. In this approach all entities in the application domain are associated to their own personal software agent. Using peer-to-peer links the system can induce, rank and recommend communication groups according to the probability that these groups are capable of handling the incident at hand. Once a recommendation is made to the requesting actor, the result is evaluated using implicit and explicit feedback mechanisms. Depending on these evaluations, the strength of the links among the agents is adjusted or new links are created.

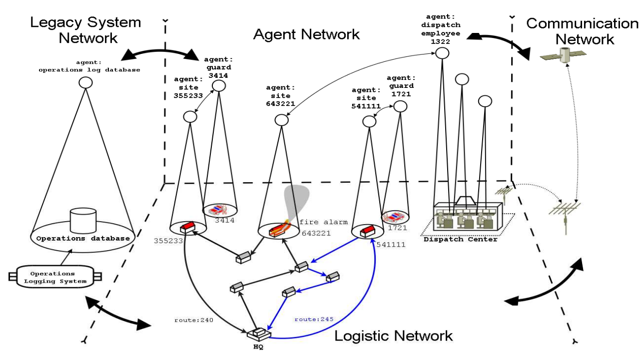


Fig 1. A schematic overview of the MHS security environment and its mapping to the agents.

While modeling the entities in the domain of MHS security, we make a distinction between those entities that are active and those that are passive (i.e., context). Active entities (e.g., guards or dispatch operators) are modeled by a set *A* of actors. We define support as the problem of finding the right person(s) $a \in 2^A$ to contact in case an alarm incident <alarm_type,o,t> $\in$ *INCIDENT* of a specific alarm type, at some security object $o_{incident}$ at a timepoint t, is notified by a dispatch operator $d \in D$:

$$SUPPORT(d, alarm\_type, o, t) = \arg\max_{a \in 2^A} v(d, alarm\_type, o, t, a) \qquad (1)$$

[1] The full version of this paper appeared in: Proceedings of the 6th German Conference on Multiagent System Technologies (Kaiserslautern, Germany, September 23 - 26, 2008).

where $v(d,alarm\_type,o,t,a) \rightarrow \mathbb{R}$ is a performance function indicating the utility of setting up communication with *a* given the incident description. While commonly most filtering algorithms are based on similarity based weighting of ratings provided by the users, we identify some additional factors that determine the utility of the support:

- Incident similarity δ^i: INCIDENT × INCIDENT → [0,1] based on expert judgments.
- Normalized feedback function φ: $D \times INCIDENT \times A \times 2^D \rightarrow [0,1]$ based on a Pearson correlation between similar dispatcher feedback weighted by incident similarity δ^i.
- Experience ε : $A \times INCIDENT\_TYPE \times O \times \mathbb{R} \times 2^O \rightarrow [0,1]$ based on the aggregation of (1) Kendall ranking correlation between handling frequency of guards at similar object, (2) the incident similarity and (3) an exponential decay function on the difference in time between two incidents as weight.
- Local rerouting cost γ: $A \times O \rightarrow [0,1]$, based normalized travel times over past incidents.

Analogous to recommendation system techniques, which are usually applied to recommend, for instance, books, movies or music to users, we propose a new approach that ranks and recommends particular communication pairs/groups – comprising *n-best* actors - to those that need assistance:

Algorithm: SUPPORTALARMINCIDENT (request_support) → setup_incident_call

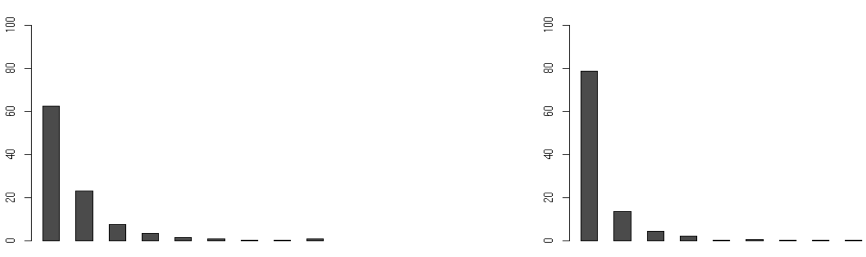
description*: this algorithm determines what call should be setup up, when an request for support for handling an alarm incident comes in.*

input: request_support($a_{incident}$,<$q_{incident}$,$o_{incident}$,$t_{incident}$>,t_0):

output: setup_incident_call($a_{incident}$, <$q_{incident}$,$o_{incident}$,$t_{incident}$> ,{a_1,…,a_n}, t_{setup})

```
candidate[] <- ∅
for ∀a'∈ A | a' ≠ a_incident do
  candidate[a'] = φ(a_incident , <q_incident,o_incident,t_incident>,a',D')
  candidate[a'] *= ε (a', q_incident,o_incident, decay, O')
  candidate[a'] *= γ(a', o_incident)
end for
candidate_sorted ← sort(candidate) by value
group_to_call ← ∅
for j = 0; j < n-best; j++ do
   group_to_call ← group_to_call ∪ candidate[j]
end for
return <a_incident,<q_incident,o_incident,t_incident>,group_to_call,t_now>
```

We evaluated the potential of our filtering approach, which gave us some promising preliminary results. In the setting without any support, 62.45% of the alarm incidents is handled within the time limit of 30 minutes.



Real distributions of alarm handling times   Simulated result when using Ask-Assist

Fig 2. Distributions of the alarm handling times.

By simulating the security domain based on a part of the logged data (n-best=3, decay=90 days), we found that for a bootstrapping period of 6 months, 72.2% could arrive within 30 minutes. Training the system on 9 months of log data gives a simulated result of 78.78%(see figure 2, right side).