A Notification Model for Smart-M3 Applications

Ivan V. Galov, Dmitry G. Korzun

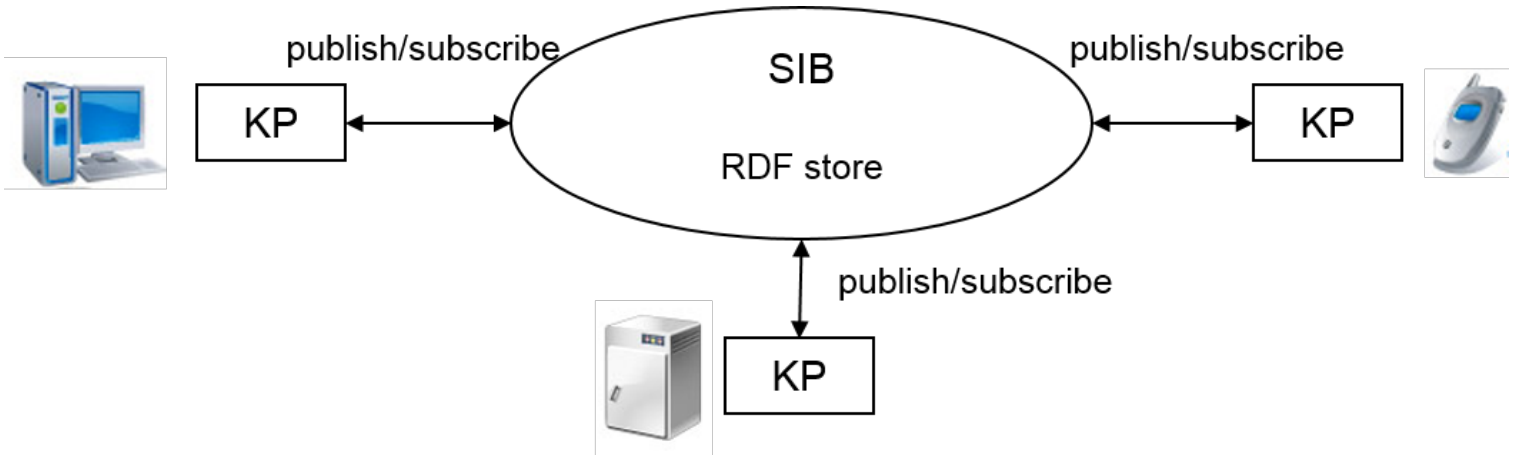
Petrozavodsk State University
Department of Computer Science

ruSMART'2014 Conference, August 28, St.-Petersburg, Russia

Table of Contents

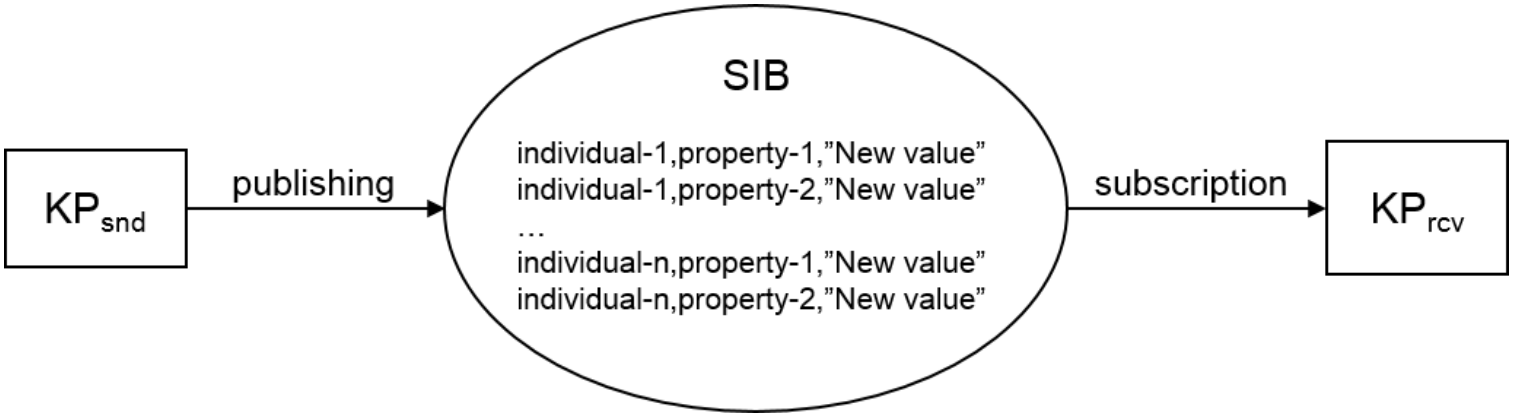
Smart Spaces and Smart-M3 Platform

- Semantic information brokers (SIBs) maintain smart space content in low-level RDF triples
- Application consists of several knowledge processors (KPs) running on various devices
- An agent sharing ad-hoc knowledge across numerous domains
  - join, leave
  - insert, update, remove
  - (un)subscribe
- Interaction between KPs is implemented using publish/subscribe mechanism



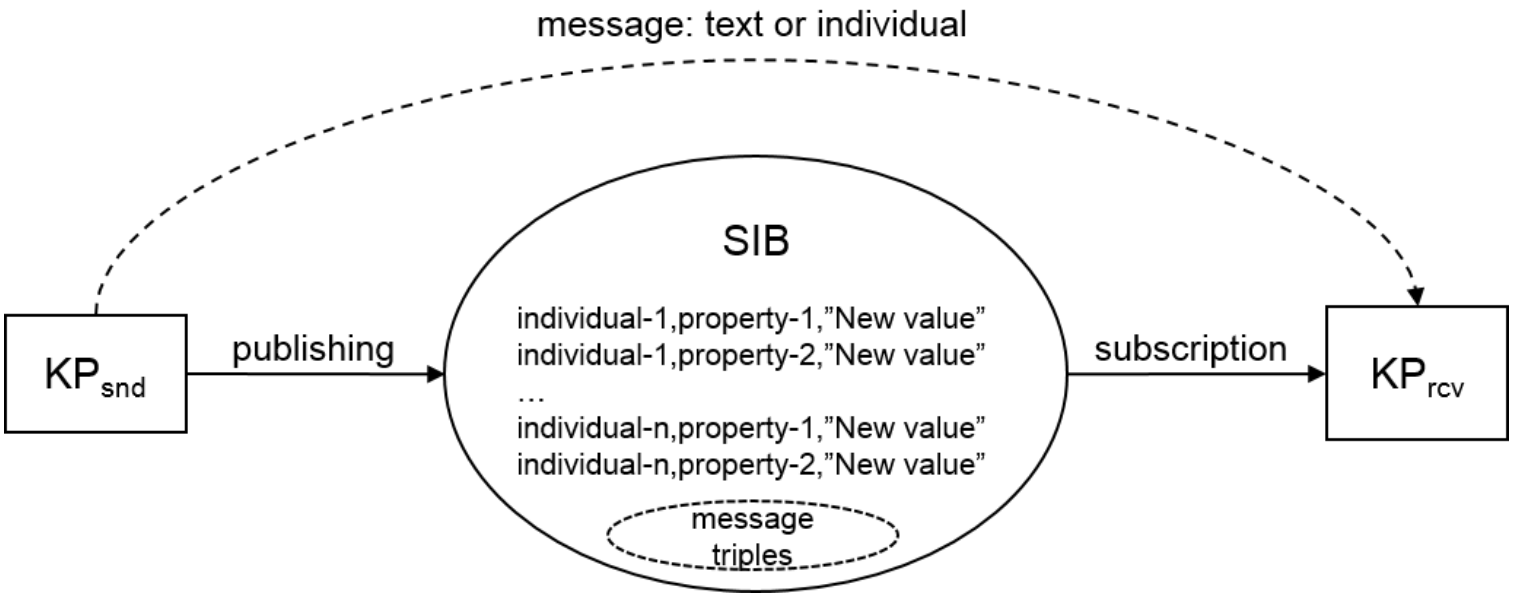
Interactions in Smart Spaces: Problem

Transfer some textual data or individual using subscription mechanism



- subscribe on each individual: too many subscriptions
- subscribe on separate properties: each individual must be updated in separate single transaction
- add "trigger" property when updated data can be retrieved: not unified (is customized for particular application)

Interactions in Smart Spaces: Solution Requirements



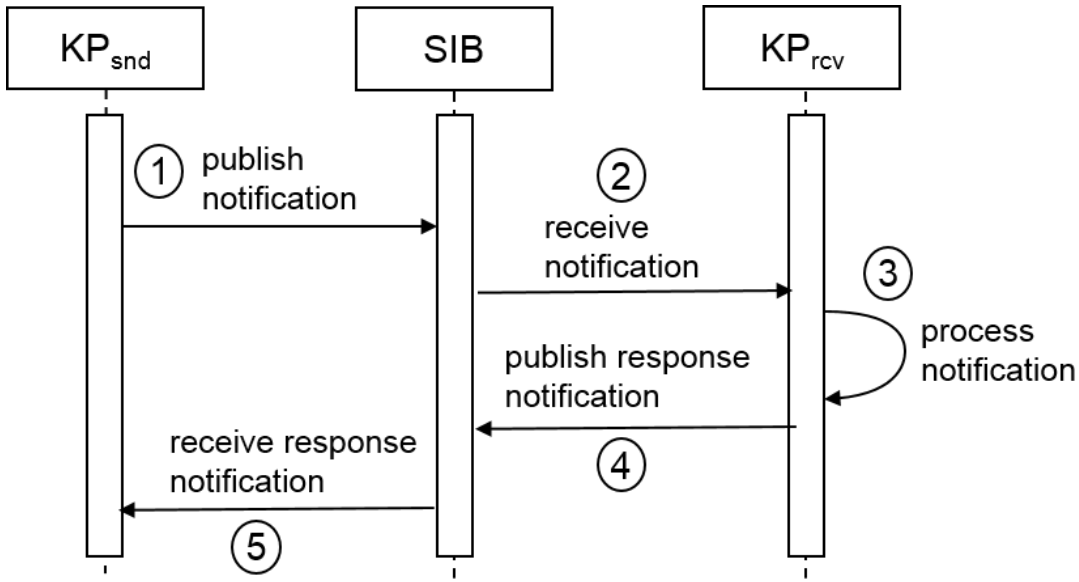
- initiate another KP (or several KPs) to perform some action (e.g., retrieve individual)
- transfer the whole individual(s) separately
- unified and extensible approach for interaction implementation
- low number of subscriptions (resource-intensive operation)

Notification Model

Notification — an informational message to be sent by KP sender ($\mathrm{KP}_{\mathrm{snd}}$) and to be received by KP receiver ($\mathrm{KP}_{\mathrm{rcv}}$) if these KPs need to attain required interaction.

Request: $\mathrm{KP}_{\mathrm{rcv}}$ performs a given operation (service) based on data provided by $\mathrm{KP}_{\mathrm{snd}}$.

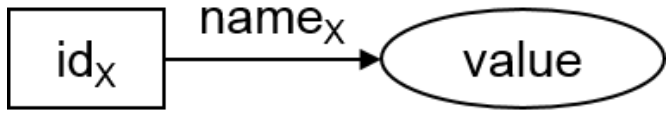
Event: $\mathrm{KP}_{\mathrm{rcv}}$ reacts on a particular event (informational fact) that $\mathrm{KP}_{\mathrm{snd}}$ is disseminating.



Notification Design Properties

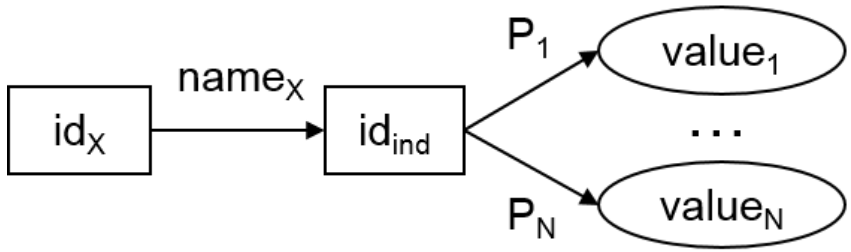
- representation: x – notification, p_i – parameter ($i = 1..n$)
  simple notification

$$\langle \mathrm{id_x} \rangle, \mathrm{rdf{:}type}, \langle \mathrm{class_x} \rangle$$
$$\langle \mathrm{id_x} \rangle, \langle \mathrm{name_x} \rangle, \langle \mathrm{value} \rangle$$



  compound notification

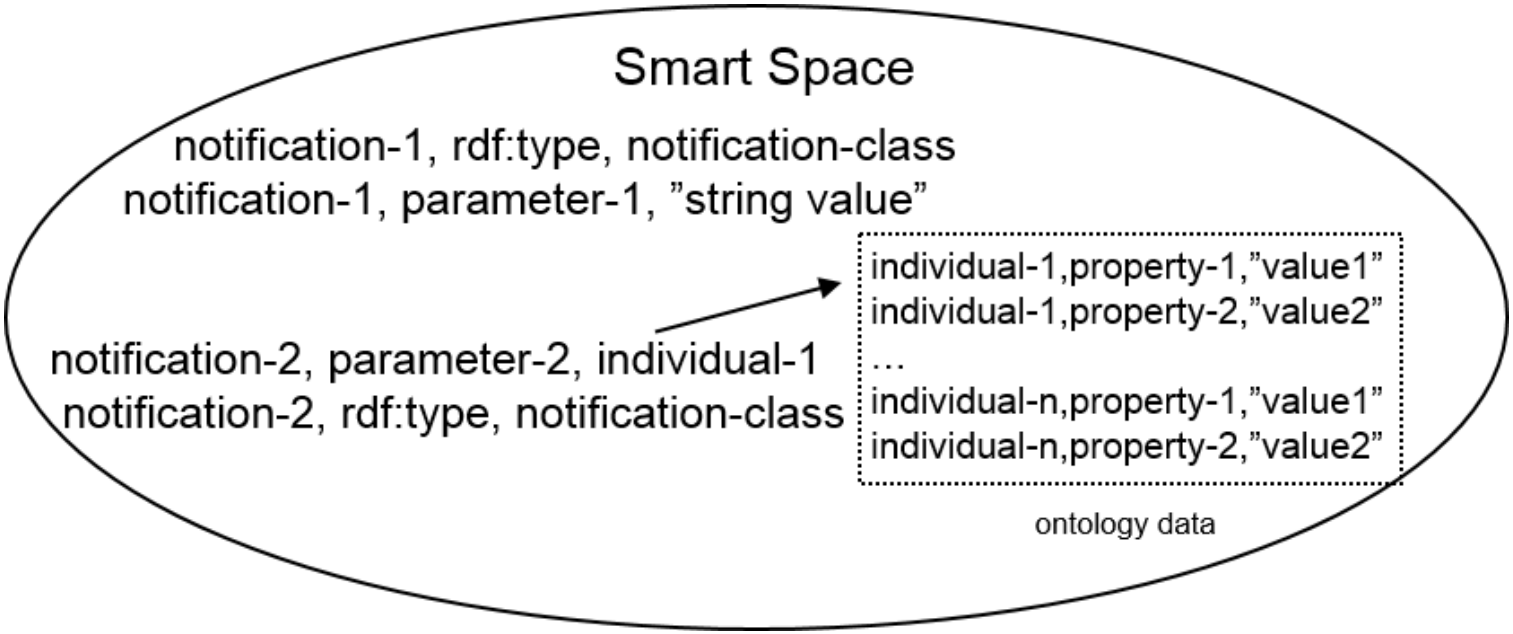
$$\langle \mathrm{id_x} \rangle, \mathrm{rdf{:}type}, \langle \mathrm{class_x} \rangle$$
$$\langle \mathrm{id_x} \rangle, \langle \mathrm{name_x} \rangle, \langle \mathrm{id_{ind}} \rangle$$
$$\langle \mathrm{id_{ind}} \rangle, \langle \mathrm{p_1} \rangle, \langle \mathrm{value_1} \rangle$$
$$\ldots$$
$$\langle \mathrm{id_{ind}} \rangle, \langle \mathrm{p_n} \rangle, \langle \mathrm{value_n} \rangle$$



- activation: reactive or proactive
- function: request or event
- response: yes or no
- clearing

Notification Ontology

Notification classes with properties–parameters



Subscription on:

$\langle \mathrm{id_x} \rangle, *, *$ (one class)

or

$*, \mathrm{rdf{:}type}, \langle \mathrm{class_x} \rangle$ (several classes)

KP Templates

Algorithm 1 $\mathrm{KP}_{\mathrm{snd}}$

1: wait for control action
2: send x (simple or compound) according to action

Algorithm 2 $\mathrm{KP}_{\mathrm{rcv}}$: simple notification

1: wait for subscription
2: get id_x from subscription
3: query parameter value:
id_x, $name_x$, $*$

Algorithm 3 $\mathrm{KP}_{\mathrm{rcv}}$: compound notification

1: wait for subscription
2: get id_x from subscription
3: query additional individual id_{ind}:
id_X, $name_x$, $*$
4: query parameters:
id_{ind}, $*$, $*$

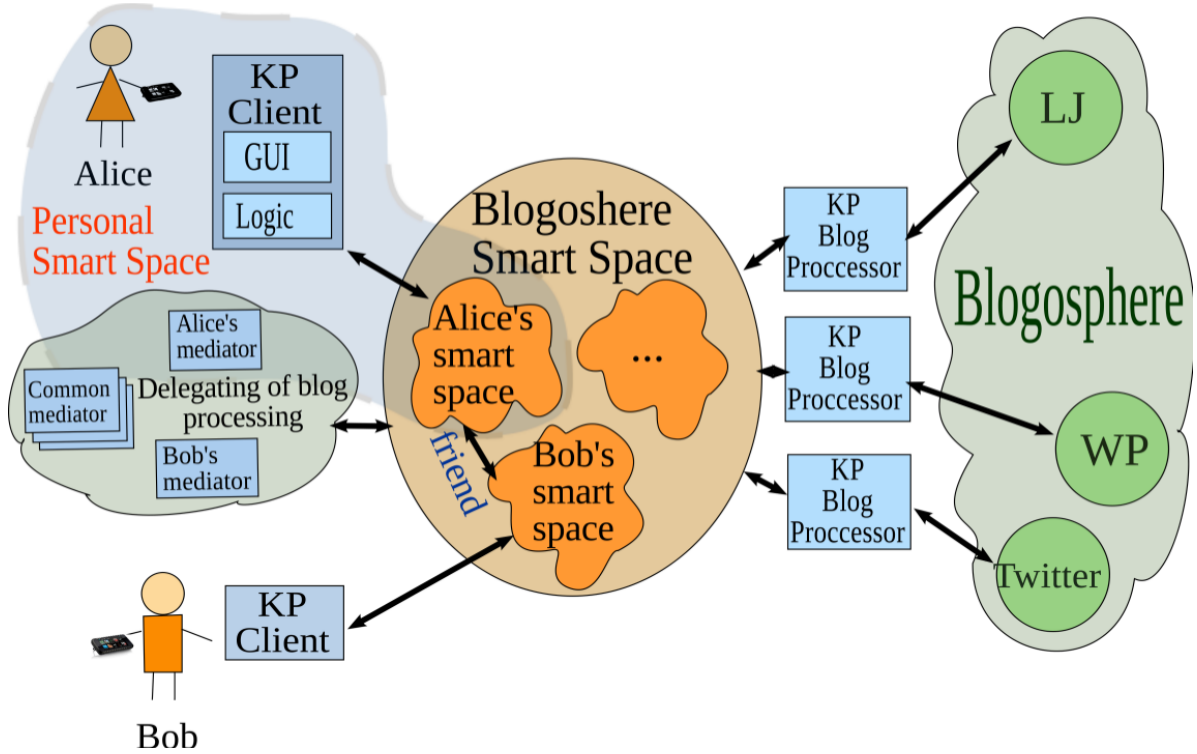
OR

5: **for** i = 1 to n **do**
6: query: id_{ind}, p_i, $*$
7: **end for**

For each interaction type — separate notification.

Case Study: SmartScribo System

- Multi-user and mobility: many clients run on mobile devices (KP clients)
- Multi-blogging: many blog services and accounts (KP blog processors)
- User collaboration: processing smart space data (KP mediators)



$$KP_{client} \leftrightarrow KP_{blog\ processor}$$

SmartScribo Notifications

1. KP client sends notification after user action
2. KP blog processor (KP-BP) receives notification on subscription
3. KP-BP performs requested operation
4. KP-BP removes notification and sends response notification
5. KP client receives and removes response

Notification	Parameters
refreshAccount	account individual
refreshPosts	account individual
sendPost	account individual, post individual
editPost	old post individual, new post individual
delPost	account individual, post individual
refreshComments	account individual
sendComment	account individual, comment individual, parent individual
delComment	account individual, comment individual, parent individual

SmartScribo Notification Examples

refreshPosts notification:

$$\text{Notification-}\langle\text{service}\rangle, \text{refreshPosts}, \langle\text{account\_id}\rangle$$

sendPost notification:

$$\begin{aligned}&\text{Notification-}\langle\text{service}\rangle, \text{sendPost}, \langle\text{notif\_ind}\rangle\\ &\langle\text{notif\_ind}\rangle, \text{postAcc}, \langle\text{account\_id}\rangle\\ &\langle\text{notif\_ind}\rangle, \text{postId}, \langle\text{post\_id}\rangle\end{aligned}$$

subscription on:

$$\text{Notification-}\langle\text{service}\rangle, *, *$$

Performance Evaluation

$t(n)$ — time elapsed since sending a notification with n parameters from $\mathrm{KP}_{\mathrm{snd}}$ until receiving it at $\mathrm{KP}_{\mathrm{rcv}}$

Two kinds of experiments:

1. retrieving all parameters in one query

$$\langle \mathrm{id}_{\mathrm{ind}} \rangle, *, *$$

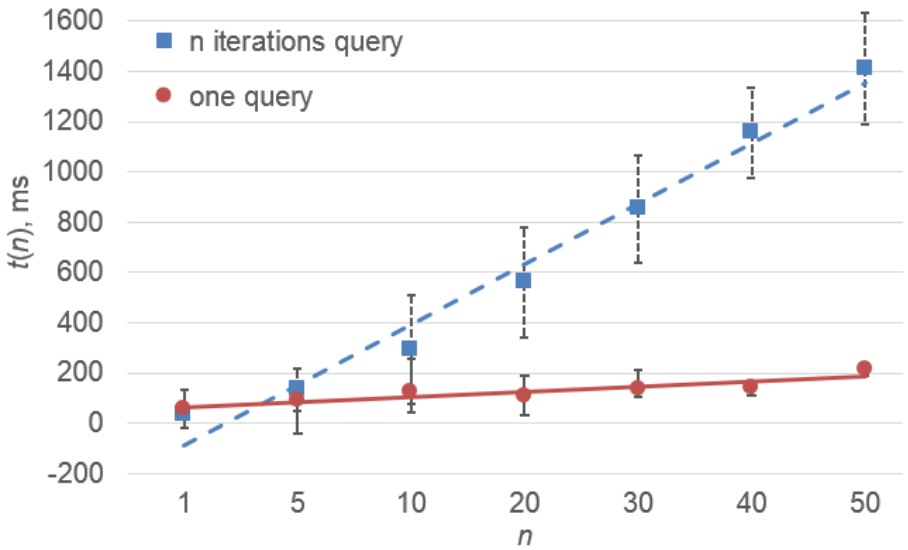
2. each parameter $i = 1, 2, \ldots, n$ is received with a separate query (n iterations query)

$$\langle \mathrm{id}_{\mathrm{ind}} \rangle, \langle \mathrm{p_i} \rangle, *$$

Measurements

- SIB running on a server machine
  (Linux OS, Intel Xeon 2.30GHz, 4GB RAM)
- $\mathrm{KP}_{\mathrm{snd}}$ and $\mathrm{KP}_{\mathrm{rcv}}$ (Python) on the same host computer
  (Windows OS, Intel Core 2 Quad 2.40GHz, 8GB RAM)
- RTT = 3 ms (SIB and KPs are in different LANs)
- measurement cycle:
  1. $\mathrm{KP}_{\mathrm{snd}}$ sends an n-parameter notification to SIB
  2. $\mathrm{KP}_{\mathrm{rcv}}$ receives the notification, retrieves values of all n parameters and removes the notification
- 100 samples for each n

Experimental Behavior



- The values fit well to linear regression
- Grows of slopes of linear regression indicates that there is some non-linear effect when *n* increases
  (slope coefficient: 27,9 (n = 100) and 23,9 (n = 10))
  - search algorithm complexity at the SIB side
  - data transfer resources the network provides to KP

Conclusion

- Notification model coordinates KPs interaction.
- The model extends the application ontology with possible requests and events.
- The model is applicable for almost any Smart-M3 application (unified approach).
- Key design properties of notification: representation, activation, function, response, clearing.
- Case study — SmartScribo system.
- Performance evaluation of model.

E-mail: galov@cs.karelia.ru

Thank you for attention!